AD-A033 653 NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER WASH--ETC F/G 15/6
THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK) PRO--ETC(U)
NOV 76 R L PAGE, A M PELLICCIOTTO

UNCLASSIFIED NMCSSC-CSM-MM-9-74-V3-CH- NL

1 OF 2
ADA033653

ADA033653

DEFENSE COMMUNICATIONS AGENCY
COMMAND AND CONTROL
TECHNICAL CENTER
WASHINGTON, D. C. 20301

(12)

IN REPLY REFER TO: C314

1 November 1976

TO: RECIPIENTS

SUBJECT: Change 1 to Program Maintenance Manual CSM MM 9-74, Volume III, Weapon Allocation Subsystem

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

3. When this change has been posted, make an entry in the Record of Changes.

4. As of January 1, 1976, the National Military Command System Support Center (NMCSSC) was reorganized as the Command and Control Technical Center (CCTC). Therefore any reference to the NMCSSC is now a reference to the CCTC in this and subsequent documentation.

FOR THE DIRECTOR

157 Enclosures
Change 1 pages

Richard L. L'Hommedieu, Ltc
for J. DOUGLAS POTTER
Assistant to the Director
for Administration

Changes to Volume;
follow cat entries
of basic reptrs
including source few series

[illegible]
3 Jan 78

EFFECTIVE PAGES - JUNE 1976

This list is used to verify the accuracy of CSM MM 9-74 Volume III after change 1 pages have been inserted. Original pages are indicated by the letter O, and change 1 pages by the numeral 1.

Page No.	Change No.	Page No.	Change No.
Front Cover	1	264	1
Title Page	1	265-272	0
ii-iii	1	273-274	1
iv-v	0	275-290	0
vi-vii	1	291	1
viii	0	292-300	0
ix-x	1	301-302	1
xi-xii	0	303-305	0
xiii	1	306-307	1
xiv	0	308-309	0
1-150	0	310	1
151	1	311-316	0
152-155	0	317-318	1
156	1	318.1-318.2	1
157-187	0	319-320	1
188-189	1	321-324	0
190-201	0	325	1
202	1	326-334	0
203-206	0	335-336	1
207	1	337	0
208-212	0	338-340	1
213	1	341-345	0
214	0	346-351	1
215	1	351.1-351.2	1
215.1-215.2	1	352	0
216-217	1	353	1
218-219	0	354	0
220	1	355-356	1
221-235	0	357-358	0
236-239	1	359-360	1
239.1-239.2	1	361	0
240-243	1	362-364	1
244-246	0	364.1-364.2	1
247	1	365-366	1
248-255	0	367	1
256-257	1	368	1
258	0	368.1-368.22	1
259	1	369-370	0
260-263	0	371	1

Page No.	Change No.
372	0
373	1
373.1-373.2	1
374-375	1
376-377	0
378-384	1
384.1-384.2	1
385-386	0
387	1
387.1-387.2	1
388-490	0
491	1
492	0
493-494	1

N
M
C
S
S
C

DEFENSE
COMMUNICATIONS
AGENCY

COMPUTER SYSTEM MANUAL
CSM MM 9-74
VOLUME III
1 JUNE 1974

NATIONAL
MILITARY
COMMAND
SYSTEM
SUPPORT
CENTER

THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK)

PROGRAM MAINTENANCE MANUAL

CH-1

(14) NMCSSC-CSM-MM-9-74-V3-change-1

NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER

(9) Computer System Manual, CSM MM 9-74

(11) 1 ~~June~~ Nov '76 1974 (12) 156p.

(6)

THE NMCSSC QUICK-REACTING GENERAL WAR

GAMING SYSTEM

(QUICK)

Program Maintenance Manual.

Volume III. Weapon Allocation Subsystem. Change 1.

(10) Richard L./Page, Angelo M./Pellicciotto, Dale J./Sanders

Submitted By:

Michael A. Blackledge

MICHAEL A. BLACKLEDGE
Major, USAF
Project Officer

REVIEWED BY:

R E Harshbarger

R. E. HARSHBARGER
Technical Director
NMCSSC

APPROVED BY:

J C Wilson

J. C. WILSON
Captain, USN
Commander, NMCSSC

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.

Approved for public release; distribution unlimited.

243000 bpg

ACKNOWLEDGMENT

This document was prepared under the direction of the Chief for Development and Analysis, NMCSSC, in response to a requirement of the Studies, Analysis, and Gaming Agency, Organization of the Joint Chiefs of Staff. Technical support was provided by System Sciences Inc. under Contract Number DCA 100-73-C-0058. Change 1 was prepared by System Sciences, Inc. under Contract Number DCA 100-75-C-0019.

CONTENTS

Section		Page

Section Page

ILLUSTRATIONS

ABSTRACT

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, simulate the planned events, and provide statistical output summaries. QUICK has been programmed in FORTRAN for use on the NMCSSC HIS 6080 computer system.

The QUICK Program Maintenance Manual consists of five volumes: Volume 1, Data Assembly Subsystem; Volume ~~II~~ 2, Weapon/Target Identification Subsystem; Volume ~~III~~ 3, Weapon Allocation Subsystem; Volume ~~IV~~ 4, Sortie Generation Subsystem; Volume ~~V~~ 5, Simulation Subsystem. The Program Maintenance Manual complements the other QUICK Computer System Manuals to facilitate maintenance of the war gaming system. This volume, Volume III provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the programs and subroutines of the Weapon Allocation subsystem. The associated program listings which are dynamic and voluminous are not contained herein. However, the program listings may be obtained by arrangement with the NMCSSC QUICK Project Officer. Companion documents are:

a. USERS MANUAL
Computer System Manual CSM UM 9-74 (five volumes)
Provides detailed instructions for applications of the system

b. COMPUTER OPERATION MANUAL
Computer System Manual CSM OM 9-74
Provides instructions and procedures for the computer operators

c. TECHNICAL MEMORANDUM
Technical Memorandum TM 90-74
Provides analytical relationships and methodology discussion for users of the system

d. ANALYTICAL MANUAL (three volumes)
Computer System Manual CSM AM 9A-67
Provides a description of the system methodology for non-programmer analysts

e. FUNCTIONAL DESCRIPTION
System Planning Manual SPM FD 90-74
Provides a nontechnical description of the system for senior management personnel

THIS PAGE INTENTIONALLY LEFT BLANK

3.6 Program ALOC

PURPOSE: This program processes the user-input command cards and calls the appropriate overlays to perform the requested operation.

ENTRY POINTS: ALOC

FORMAL PARAMETERS: None

COMMON BLOCKS: ALOCIN, ASMT, CONTROL, CORRCHAR, CTRYCD, C222, C333, DEFENSE, DRC5162, DYNAMIC, FILABEL, FILES, FIXED, FIXEDASS, FORMTT, HOB, IFTPRN, INTFILE, IRUNTY, ITP, LAMBDA, LOCFIL, MACHINE, MASTER, MYIDENT, MYLABEL, MULADJ, NALLY, NOPRINT, PAYDATA, PAYLOAD, PAYOFF, PEN, PKNAVAL, PLANTYPE, PRINEED, PRIN1, PRNTCN, PRNTWADD, PRTMULL, REF, SALVO, SMAT, TABLE, TWORD, WADFINAL, WADOTX, WADWPN, WAROUT, WPNREG, WPNTYPE, WPNGRP

SUBROUTINES CALLED: IDTS, INITAPE, INITALC, MULCON*, OVAL1*, RDARRAY, SETSAL, SETREAD, SKIP, TIMEME, TERMTAPE

CALLED BY: This is the main program; entered by GCOS System

Method:

This main program processes the user-input command cards and controls the initialization routines. Subroutine INITALC is called to initialize program variables. A call on TIMEME with a parameter of -1 initializes the timing clock. Subroutine INITAPE is called to initialize the file-handler. Subroutine SETREAD is called to initialize reading on the BASFILE. INFORM (block /FILABEL/) is tested to determine if the BASFILE is of the correct format. If not the run aborts with an error message (statement 10). If the format is correct, the external common blocks MASTER, FILES, ASMT, CORRCHAR, PAYLOAD, REF, PLANTYPE, WPNREG, WPNTYPE, WPNGRP, PKNAVAL, PAYDATA, HOB and CTRYCD are filled from the BASFILE (statements 20 and following). Extraneous portions of the BASFILE are skipped over by subroutine SKIP. The number of weapons fixed in each group is read into a temporary storage area, ITEMP equivalenced to ALERREST in block/C222/, and subtracted from the number of weapons per group, NWPNS in block /WPNGRP/. The BASFILE is terminated and the time to read the file is recorded by a call on TIMEME.

The remainder of the subprogram reads and processes the user-input command cards. Each card is read at statement 100.

*These routines are called via the system loader LLINK.

The first overlay (OVAL1) is called to process the input cards for the requested function. If the requested function was not ALLOCATE, control returns to statement 100 to read the next option.

If the command was ALLOCATE, the routine determines from variable IADD in /FIXED/ whether fixed assignments were requested. If not, the fixed weapons are added back into the stockpile.

At statement 300, subroutine SETSAL is called to initialize common block /SALVO/. Overlay two (MULCON) is called to allocate the weapons.

Local variable HAVE is a parameter for the system routine LLINK. Depending on the value of KHAVE, the particular overlay is loaded (via LLINK) or called directly since it already resides in core.

Program ALOC is illustrated in figure 22.

3.6.1 Subroutine FORMATS

PURPOSE: This subroutine determines the best 10-column BCD format for a variable.

ENTRY POINTS: FORMATS

FORMAL PARAMETERS: None

COMMON BLOCKS: FORMTT

SUBROUTINES CALLED: None

CALLED BY: PRNTNOW

Method:

The variable to be formatted is INWORD, the first word in common /FORMTT/. (This word is equivalenced to WORDIN.) The best 10-column format is returned in NFORMAT, the second word of common/FORMTT/. The resulting value of NFORMAT can be used in a FORTRAN output statement such as: PRINT NFORMAT, INWORD.

The names WORDIN and INWORD are equivalenced to allow correct specification (real or integer) for any possible input. The name NFORMT is internally equivalenced to the variable name NFORMAT for convenience. Internal to FORMATS, the absolute value of the input variable is kept in INABS, equivalenced to ABSIN for type specification purposes.

To determine if a number is fixed or floating point, the first 12 bits of the absolute value are tested. If a bit is set, the number is assumed to be floating point, since all normalized floating point numbers have at least one bit set in this range. Thus, if an integer quantity greater than 16,777,216 is input, it will be treated as if it were a floating point variable. However, no variable input from PRNTNOW can have a value of that magnitude, so this restriction is never a handicap. Table 13 presents the returned formats for each range of input values.

Subroutine FORMATS is illustrated in figure 23.

Table 13. Calculated Formats for Variables

RANGE OF ABSOLUTE VALUE OF VARIABLE, X	FORMAT OF NEGATIVE	FORMAT OF POSITIVE
0 or Integer Variable	I10	I10
$0 < X < 0.0001$	E10.1	E10.2
$0.0001 \le X \le 0.999999$	F10.5	F10.5
$0.999999 < X \le 99.9999$	F10.4	F10.4
$99.9999 < X \le 9999.99$	F10.3	F10.3
$9999.99 < X \le 999999.9$	F10.2	F10.2
$999999.9 < X$	E10.1	E10.2

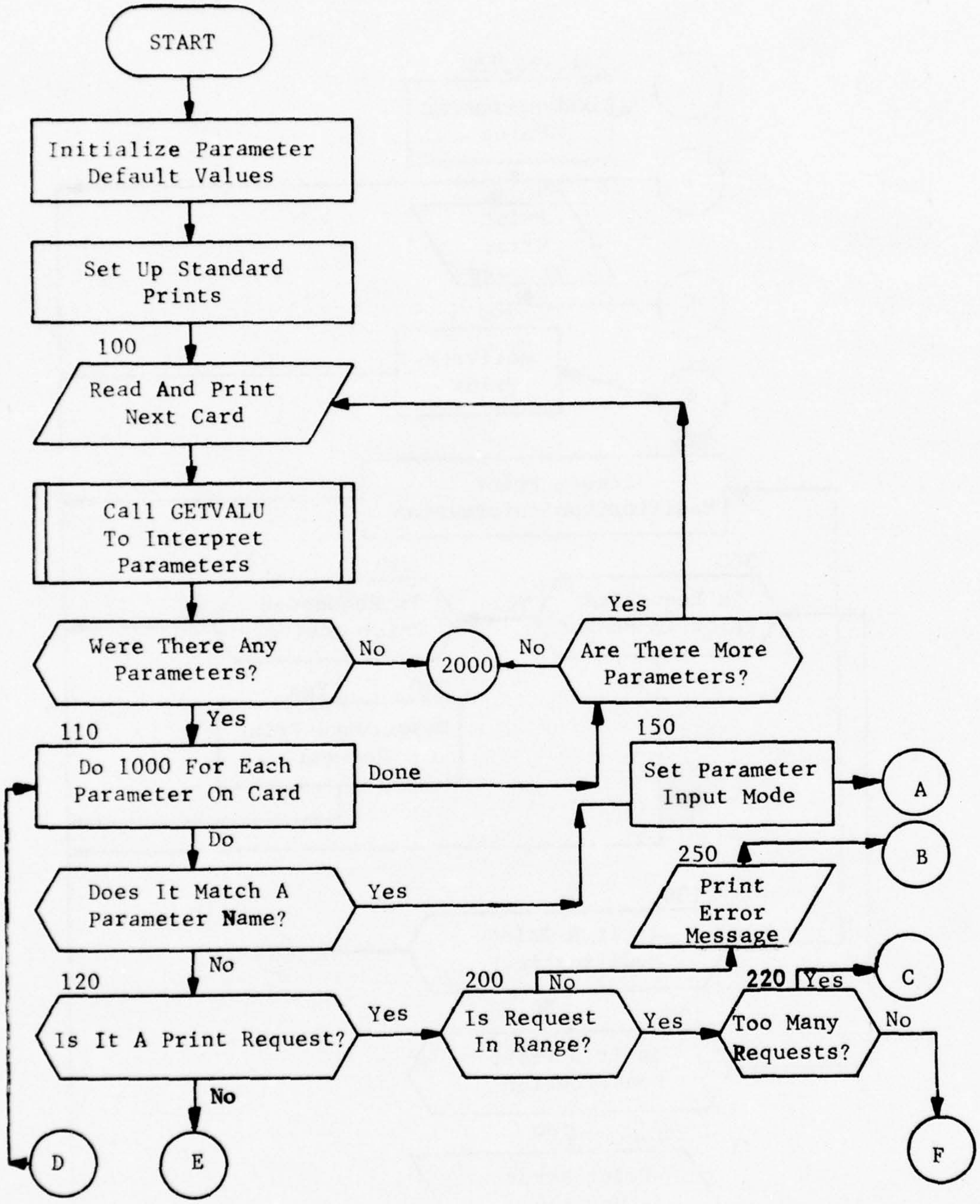


Figure 33. Subroutine RDALCRD
Part I: Input Card Processing

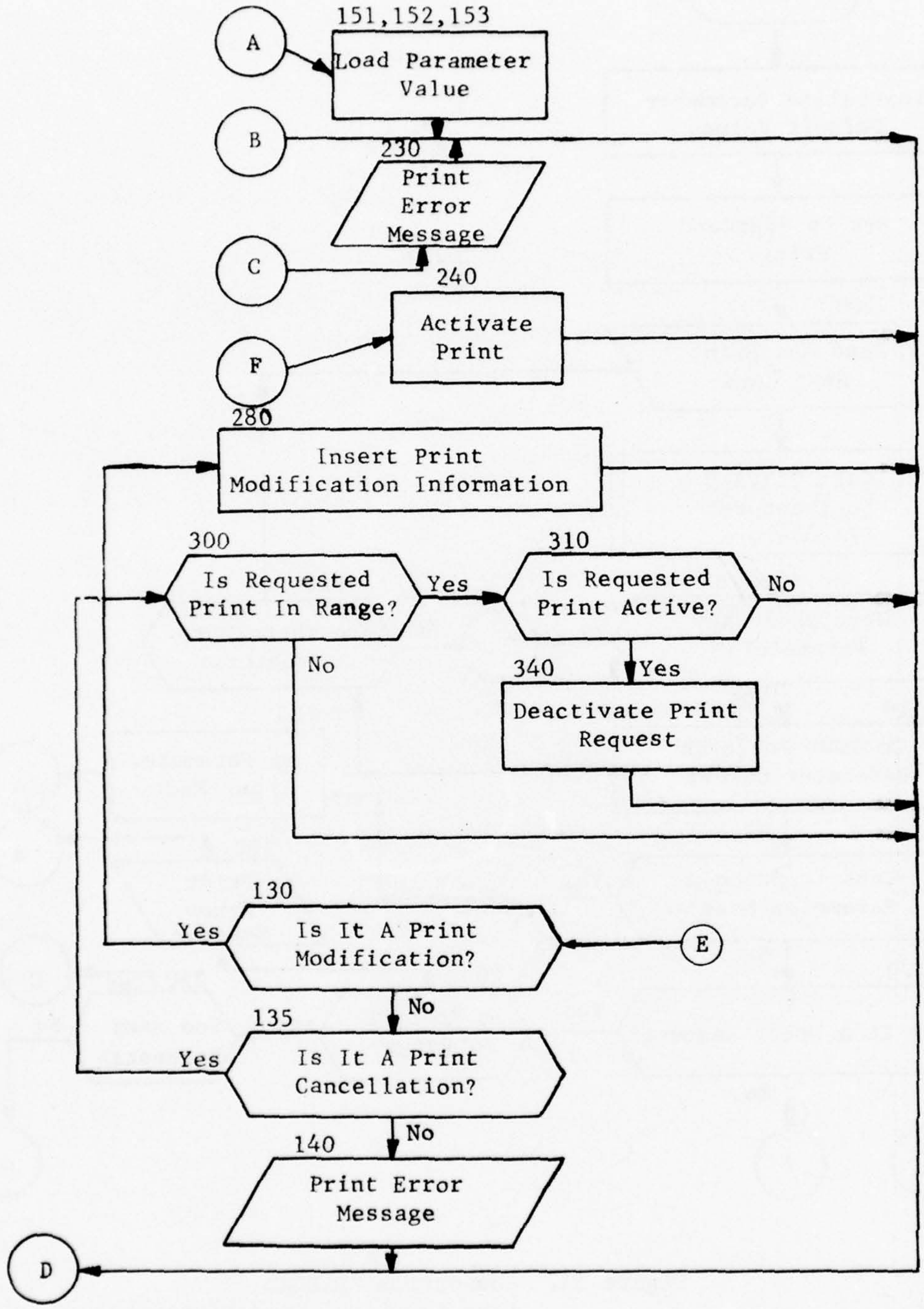


Figure 33. (Part 2 of 2)

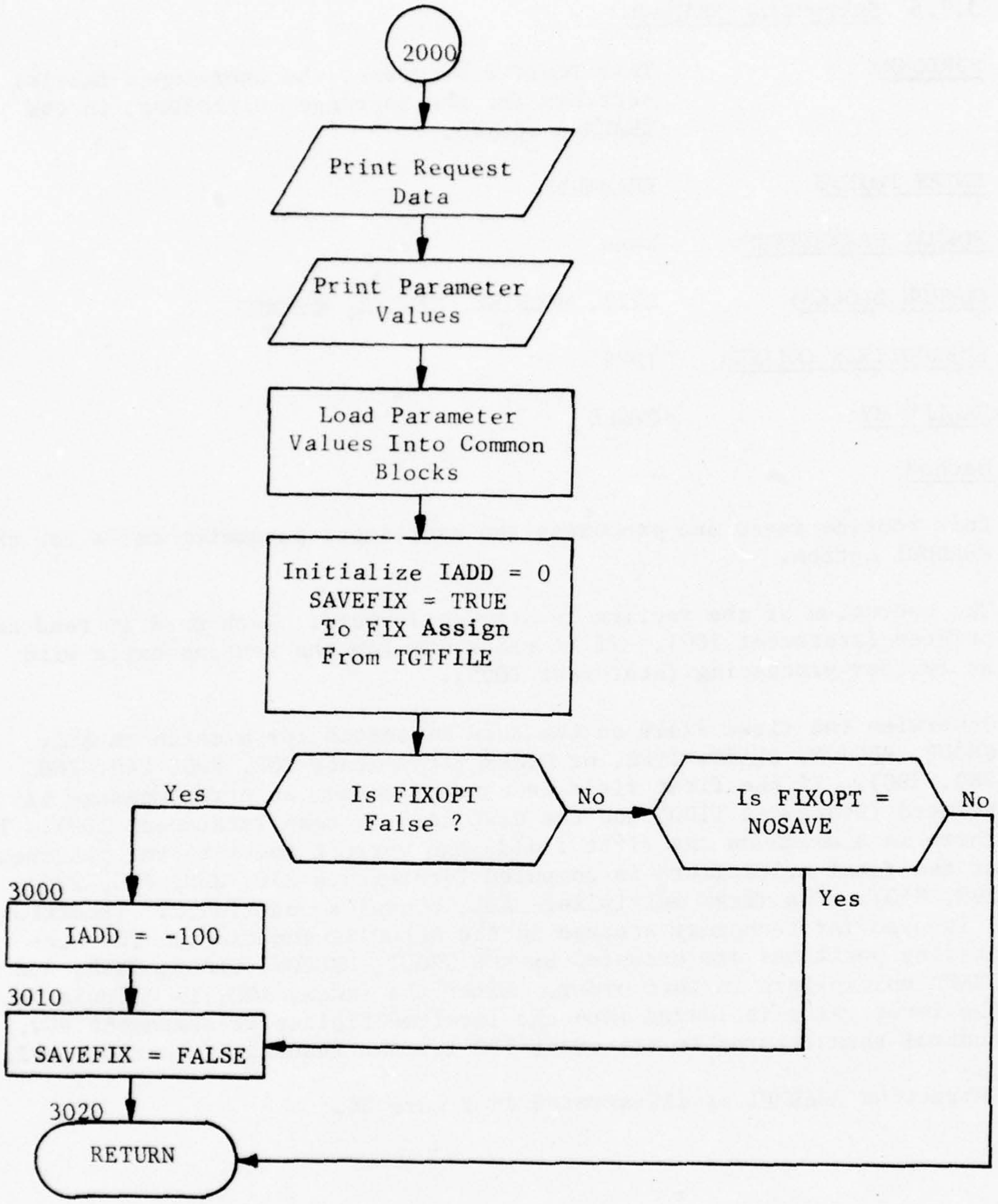


Figure 33. Part II: Post Input Processing

3.7.6 Subroutine READMUL

PURPOSE: This routine processes the user-input initial settings for the Lagrange multipliers in the READMUL option.

ENTRY POINTS: READMUL

FORMAL PARAMETERS: None

COMMON BLOCKS: C222, MACHINE, MASTER, WAROUT

SUBROUTINES CALLED: IDTS

CALLED BY: OVAL1

Method:

This routine reads and processes the user-input parameter cards for the READMUL option.

The operation of the routine is straightforward. Each card is read and printed (statement 100). If it reads END LAM the routine exits with no further processing (statement 1000).

Otherwise the first field on the card is tested for a match on ALL, GROUP, REGION, CLASS, TYPE, or OTHER (statements 200, 220, 240, 260, 280, 300). If the first field is none of these, an error message is printed (Statement 1100) and the next card is read (statement 100). If there is a match on the first field, the correct index to the placement of the local multipliers is computed (statements 210, 230, 250, 270, 290, 310). The first multiplier, ALL, occupies position 2. (Position 1 is used for temporary storage in the ALLOCATE function.) The succeeding positions are occupied by the GROUP, REGION, CLASS, TYPE, and OTHER multipliers in that order. After the index, LOU, is calculated, the input value is loaded into the local multiplier at statement 900. Control then returns to statement 100 for the reading of another card.

Subroutine READMUL is illustrated in figure 34.

During the main allocation phase PROGRESS is less than 2.0 so usually the termination closing phase on the lower right of the sheet is skipped entirely.

When PROGRESS first becomes equal to 2.0, the allocation is complete and the file then being written on is noted as STRKIN for ALOCOUT (statement 2721, Part V). However, the allocation at this point is not all on that file. Consequently, in order to get the allocation on one file, it is necessary to continue reading the old and simply copy the old allocation to fill out the new file. This is done in two ways. If verification or testing of the allocation is to be done (IVERIFY = 1, 2), the old allocation is saved in statement 1220 and the program goes on to carry out the verification, reading in the old allocation to the next target as usual in statement 1201. Since verification always requires a full pass, this process will assure that the STRKIN file is complete. (When the end of the file is reached and files are switched, it continues writing on the next file, but this does no harm.) When the verification is complete all files are terminated (statement 1231) and subroutine SORTMIS is called to write the MSLTIME file for use by program PLANOUT. A test is then made to determine if the STRKIN file is the intended file, ALOCTAR. If so, control returns to the main program immediately after a final print of timing data. If not, the ALOCT1 file is copied onto the ALOCTAR file before the timing print and return.

However, assuming that PROGRESS is not yet equal to 2.0, the termination section is ignored and the program gets ready to generate a new allocation to replace the one just read in statement 1201. On the first pass the old allocation is a pseudo-allocation and the replacement is done elsewhere (see Part II). The replacement is accomplished by removing the contribution of the old allocation* to all running sums before the new allocation is generated. This is accomplished in the 1300 series of statements. This can be done in this simply way because the values of COST, PAYOFF, PROFIT, TGTWT, and DPROFIT, then in memory, are those just read in from the dynamic file and so they correspond to the old allocation. Since the quantities RUNSUM and WTSUM were divided by WTFAC at the end of the last pass when the files were interchanged, the old TGTWT must also be divided by this same factor to make it commensurate before it is subtracted out.

The reason that REVCOST must be computed is that the values of the multipliers have probably changed (unless PROGRESS = 1.0) since the prior allocation; consequently a revised cost (REVCOST) of the allocation based on the new multipliers is of interest, and is probably

---

*The fixed weapons are ignored because they do not contribute to the running sums.

different than the old cost COST. The reason for the test on PROGRESS before correcting the cumulative differential profit will be discussed in connection with Part IV of the program (Part 5 of figure 37).

Subroutine ADDSAL is used to maintain the sums for the salvoed groups.

Part IV: Processing After Allocation

Before calling STALL,* CTSPILL is set to 0. (If some elements are spilled, WAD will so note by setting CTSPILL equal to the number of elements spilled.) The calls on TIMEME (5, 6, 7) before and after the allocation cause the time spent during the actual allocation to be recorded in columns 6 and 7 of the TIMEME output print (number 23).

Before calling either allocation routine, however, the program must check the number of fixed weapon assignments. The limitation on weapons allocated to one target is 30 weapons on an undefended target and 30 weapon groups on a defended (i.e., terminal ballistic missile defenses) target. Usually, MULCON calls both STALL* and DEFALOC* on defended targets and chooses the best allocation. If there are more than 30 fixed weapon assignments, STALL* should not be called. If the number of fixed assignments is greater than 30, MULCON checks to see if it is a defended target. If not, an error message is printed and the excess assignments are ignored. Then STALL* is called. If it is a defended target, MULCON sets a dummy low profit (except for verification) and calls only DEFALOC.*

The additional details of the allocation required by later processors are then recorded in /DYNAMIC/. PEN and TOARR are required by EVALALOC, while KORR and VTD (as changed to compute RVAL) are required by ALOCOUT, FOOTPRNT, and POSTALOC. Subroutine BOMPRM is then called to update the ASM allocation fraction array FASM.

The various running sums are then calculated in statements following 1402. If DEFALOC has made the allocation, the KORR array gives the number of missiles from each group allocated to the target. If KORR is positive, it represents the corridor. If it is negative, it represents the number allocated. Then the profit and cost data are recorded.

The following quantities are of particular importance:

DPROFIT = PROFIT - OPROFIT

SDPROFIT = Σ DPROFIT * CTMULT

DELTEFF = DPROFIT/VALWPNS

SDELTEFF = SDPROFIT/VALWPNS

*STALL and DEFALOC are called via computer system subroutine LLINK.

or

$$R_1/R_0=(\lambda_1/\lambda 0)^{-n}$$

so

$$\frac{\partial(R_1/R_0)}{\partial(\lambda_1/\lambda_0)}= -n$$

For small differences between λ_0 and λ_1 this implies:

$$\frac{R_1 - R_0}{R_0} = -n \frac{\lambda_1 - \lambda_0}{\lambda_0}$$

Solving for the new value λ_1 of λ

$$\lambda_1=\lambda_0(1 + \frac{(R_1 - R_0)/(-n)}{R_0})$$

If we now identify a new variable R_2 as the ultimately desired allocation rate, R_1 as the new rate we hope to obtain with λ_1, and R_0 as the current allocation rate -- then the above variables can be associated with information already available as follows:

$$R_1 - R_0 = \text{CORFAC*}(R_2 - R_0) = \text{CORFAC*ALERREST}$$

$$R_0 = \text{ALERREST} + (\text{NOWPS/NTGTS})$$

If we now associate the FORTRAN variable PARTIAL with n and the local multiplier LA with λ this gives rise to the following procedure for updating LA:

$$LA_1 = LA_0 * \left[1.0 + \frac{\text{CORFAC * ALERREST(J, INTPRD)/(- PARTIAL)}}{\text{ALERREST(J, INTPRD) + (NOWPS(J)/NTGTS)}}\right]$$

This formula is well behaved if ALERREST is large and positive, but if it is negative and as large as the expected rate (NOWPS(J)/NTGTS) (i.e., if the actual allocation rate is zero), then the denominator goes to 0. In this case an infinite correction would be indicated. To avoid this, the expected rate in the denominator is multiplied by 2 giving:

$$LA_1 = LA_0 * \left[1.0 + \frac{\text{CORFAC * ALERREST(J, INTPRD)/(- PARTIAL)}}{\text{ALERREST(J, INTPRD) + 2 * (NOWPS(J)/NTGTS)}} \right]$$

This is the function used in statement 2401.

In the present version of the program the value of PARTIAL(J) has been set equal to 1.0 for all the local multipliers LA(J). This choice is based on the effect of the premium on the sensitivity of the allocation rate to the value of LAMEF or λ. When the multipliers are almost correct, it is usually the case that most weapon groups are in close competition with many other groups with very similar properties. Then a small change in the multiplier LAMEF will produce a very large change in the allocation rates, as the weapon group in question almost totally replaces, or is replaced by, its competitors.

However, such a large error in the allocation rate will not actually occur because as the error builds up the estimated value of the payoff will be automatically changed by the premium. Thus, for constant values of LAMEF, when an equilibrium allocation rate is reached, it must be approximately true that the error in LAMEF is compensated by the premium. That is, if λ_0 is the correct value for LAMEF then:

$$\text{LAMEF - PREMIUM} \cong \lambda_0.$$

Since

$$\text{PREMIUM} = \text{PRM * LAMEF} * \frac{\text{SURPWP - .5 * CTMULT}}{\text{NWPNS}}$$

we can define a relation between LAMEF and (SURPWP/NWPNS).

$$\text{LAMEF} * (1 - \text{PRM} * \frac{\text{SURPWP - .5 * CTMULT}}{\text{NWPNS}}) \cong \lambda_2$$

Since this relationship is the same for all groups it is reasonably simple to use the same value 1.0 of partial derivative for all local multipliers.

In the 2500 series of statements the values of LAMEF(G) are recomputed using the new values of the local multipliers LA(J). At the same time it is necessary to reevaluate the summation of the value of all the weapons VALWPNS = Σ LAMEF(G) * NWPNS(G) and the summation of the value of the error in weapons allocated.

VALERR = Σ *LAMEF(G) * ABSF(SURPWP(G)) using the updated values of LAMEF.*

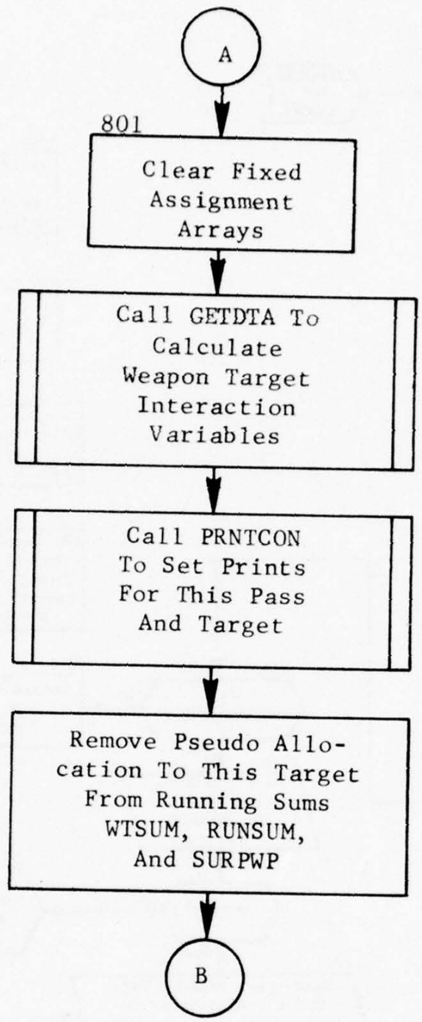


Figure 37. Part II: Fixed Weapon Assignment Processing

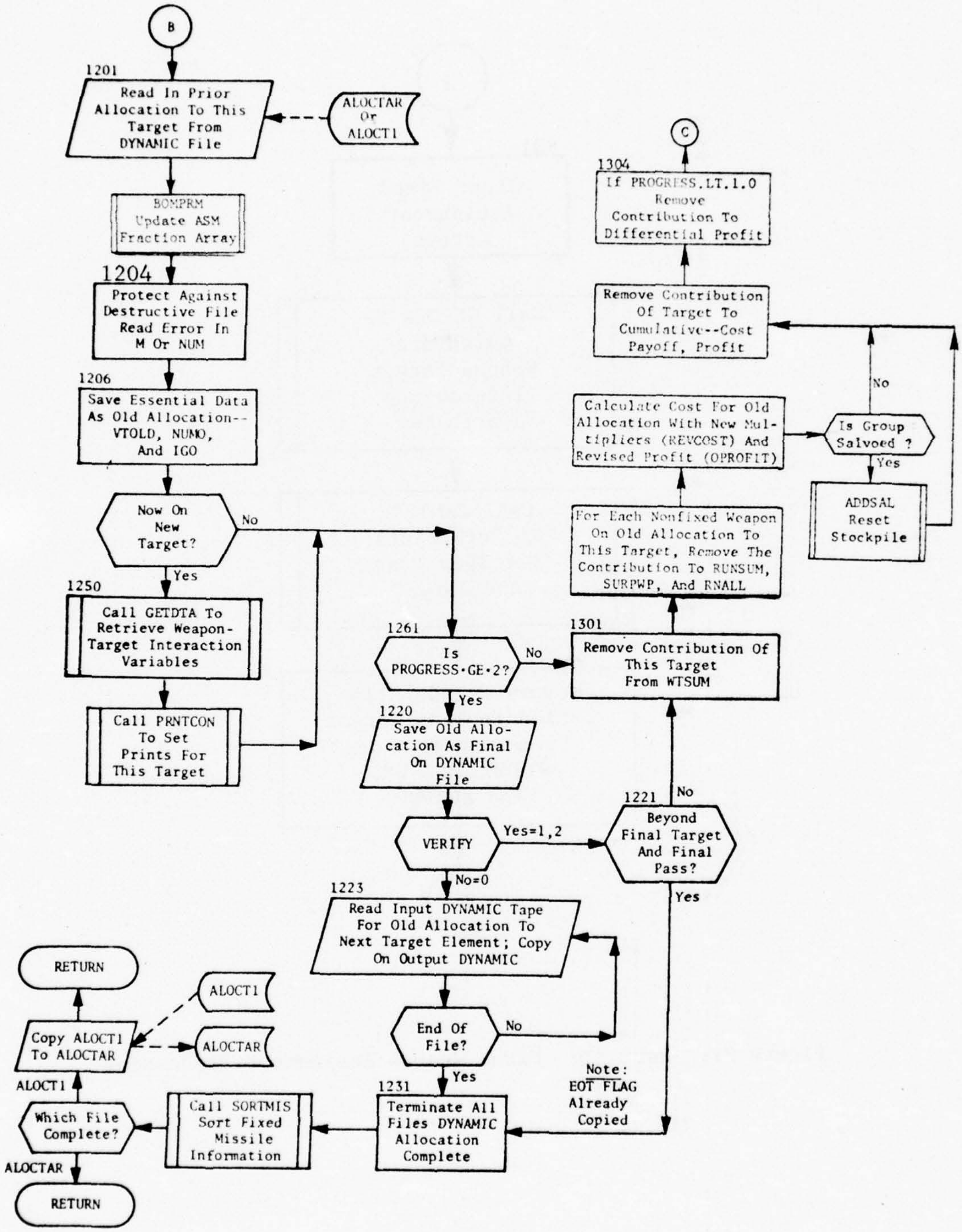
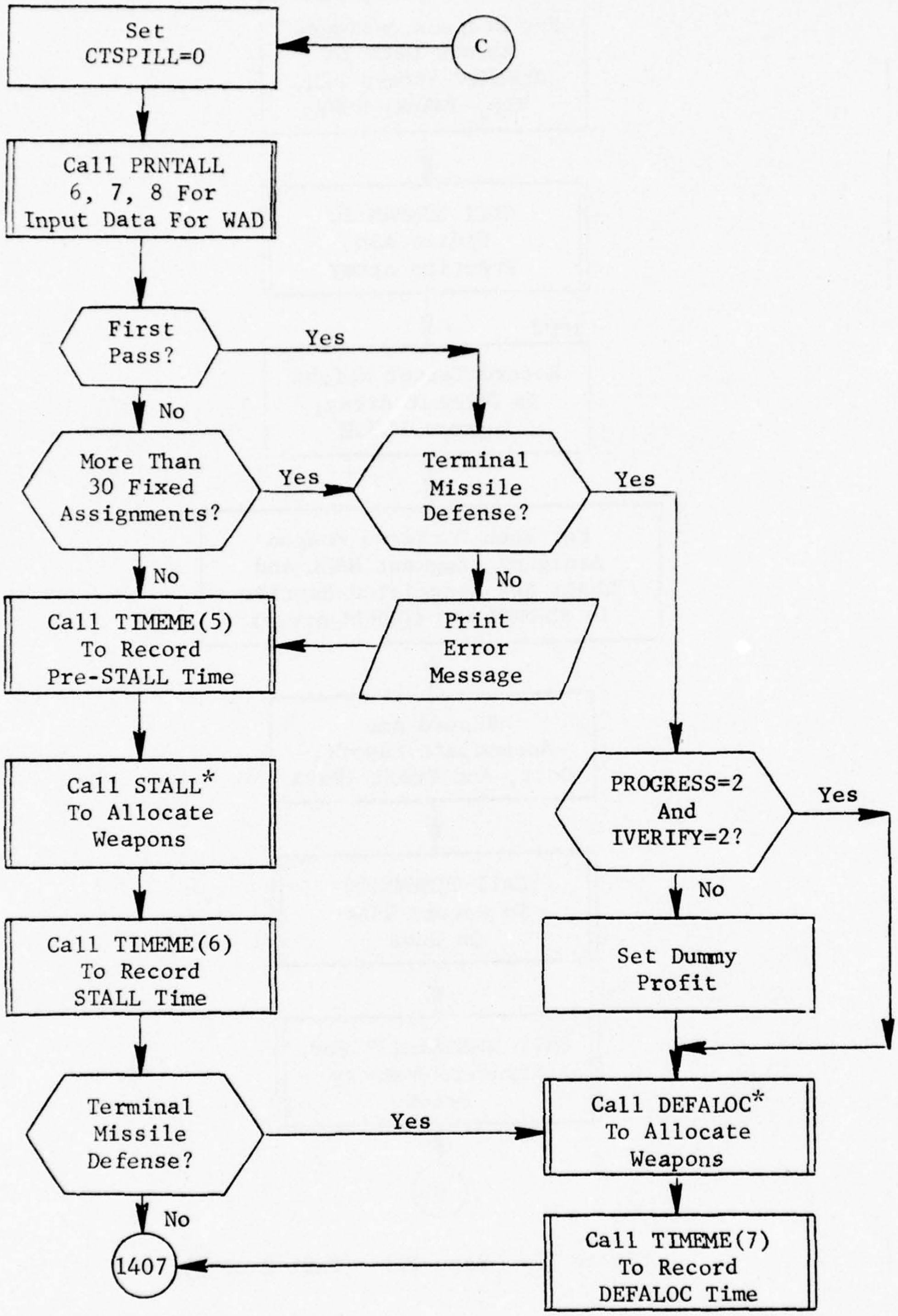


Figure 37. Part III: Main Flow (After First Pass)



*Called via computer system subroutine LLINK.

Figure 37. Part IV: Processing After Allocation
(Part 1 of 5)

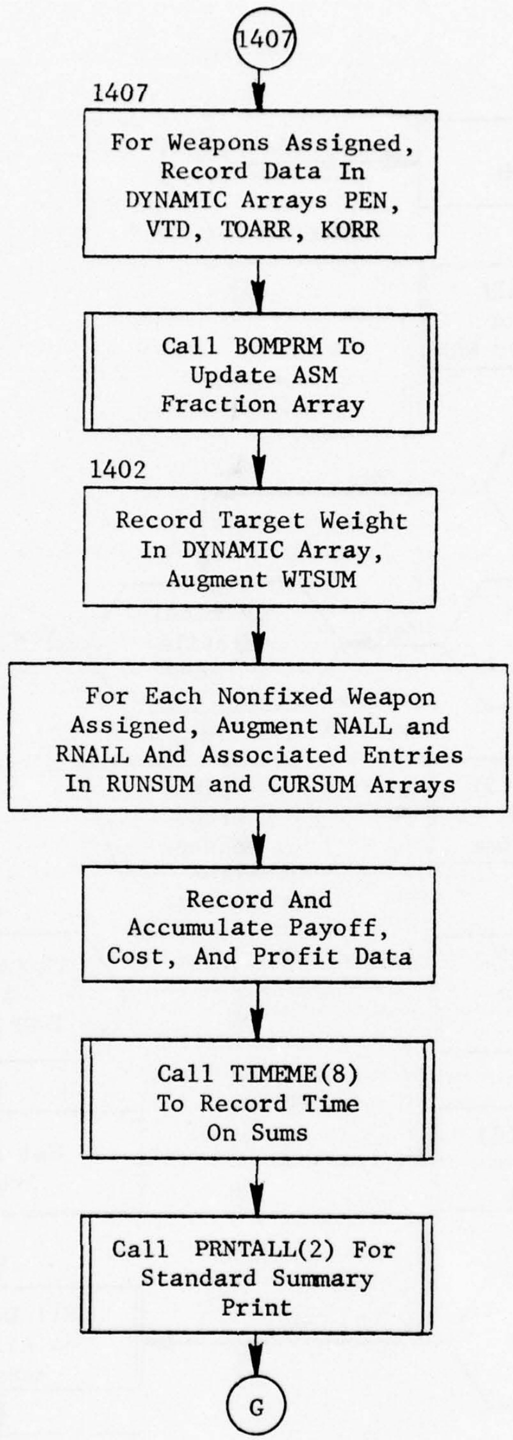


Figure 37. Part IV: (Part 2 of 5)

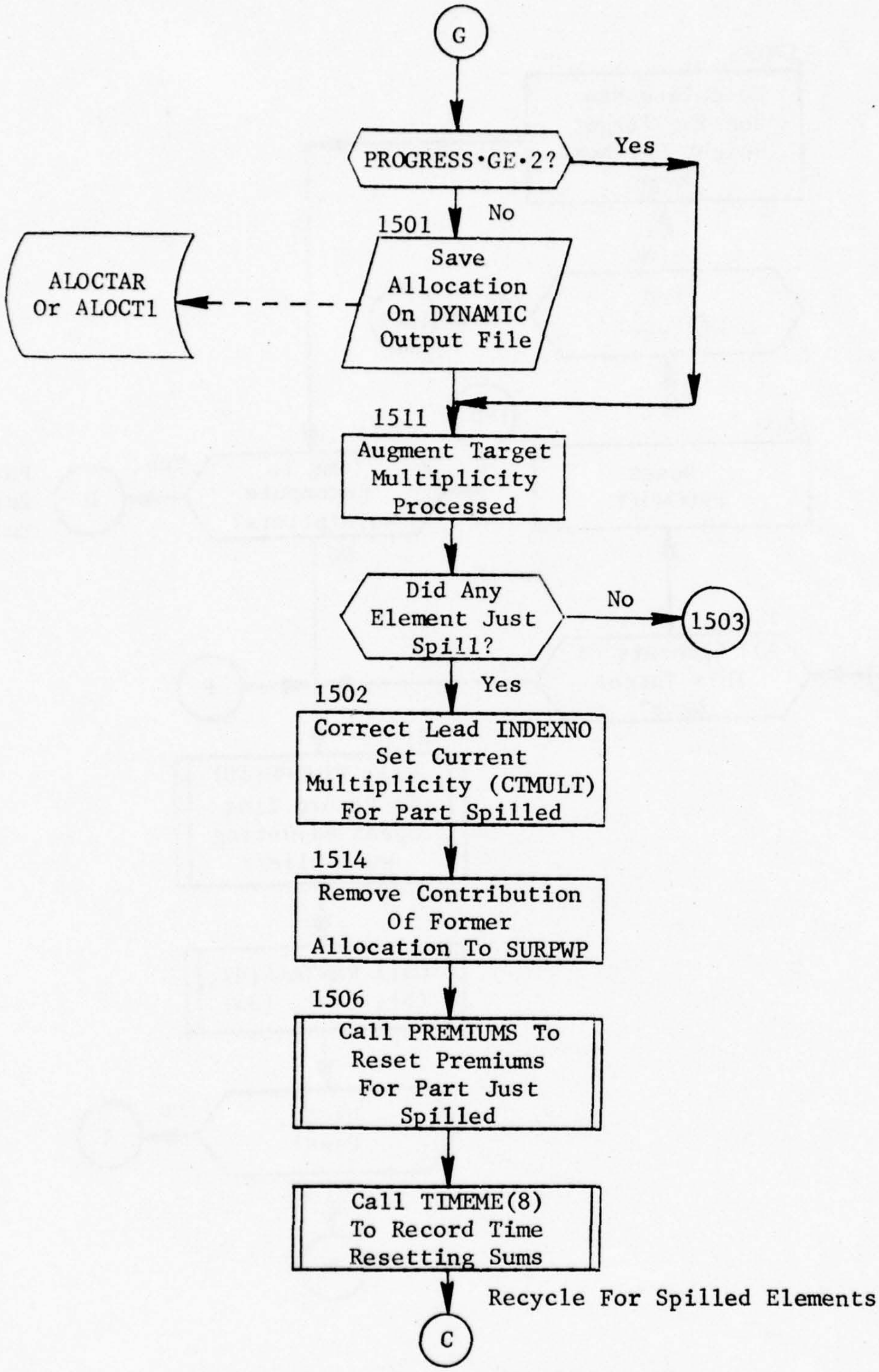


Figure 37. Part IV: (Part 3 of 5)

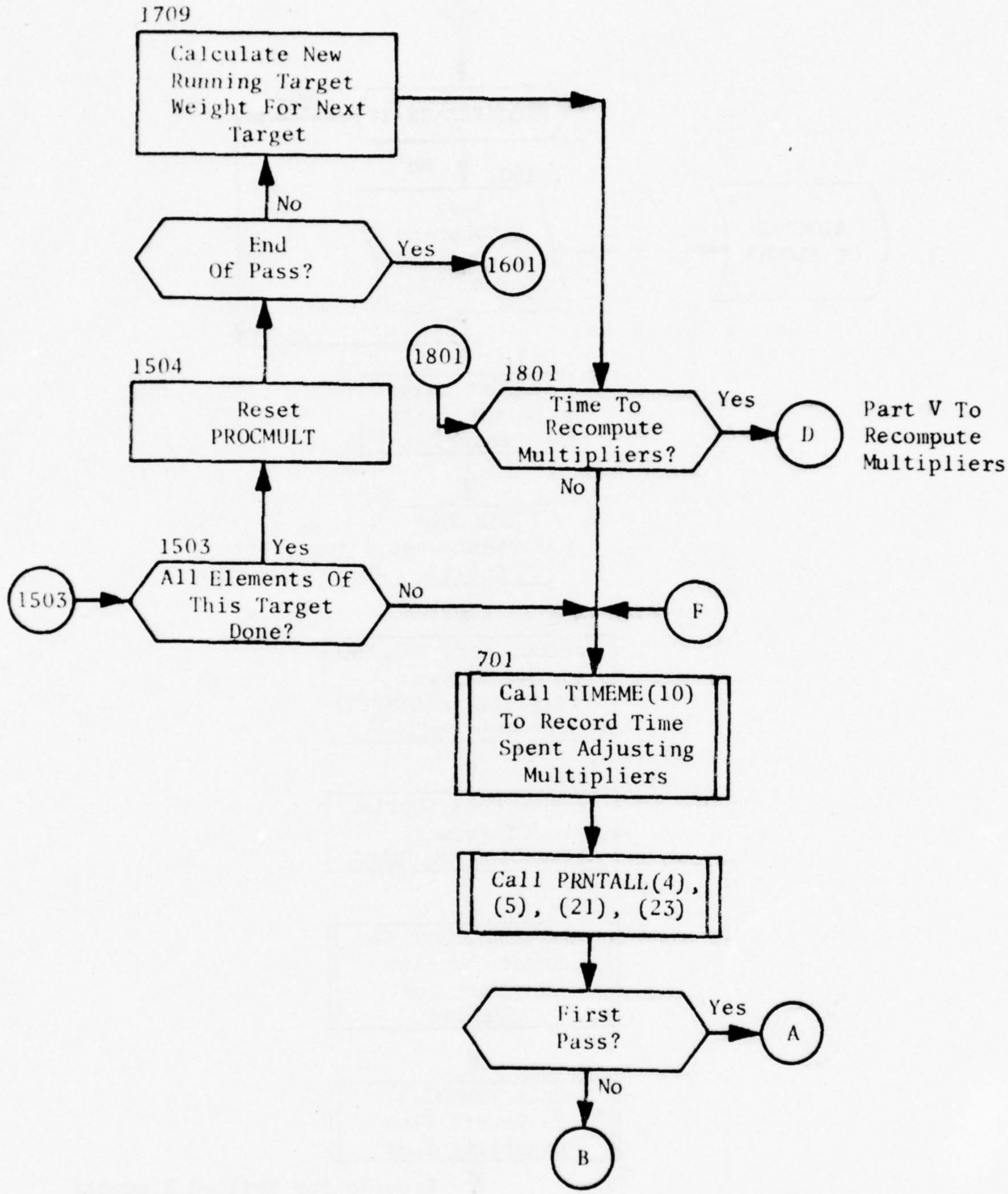


Figure 37. Part IV: (Part 4 of 5)

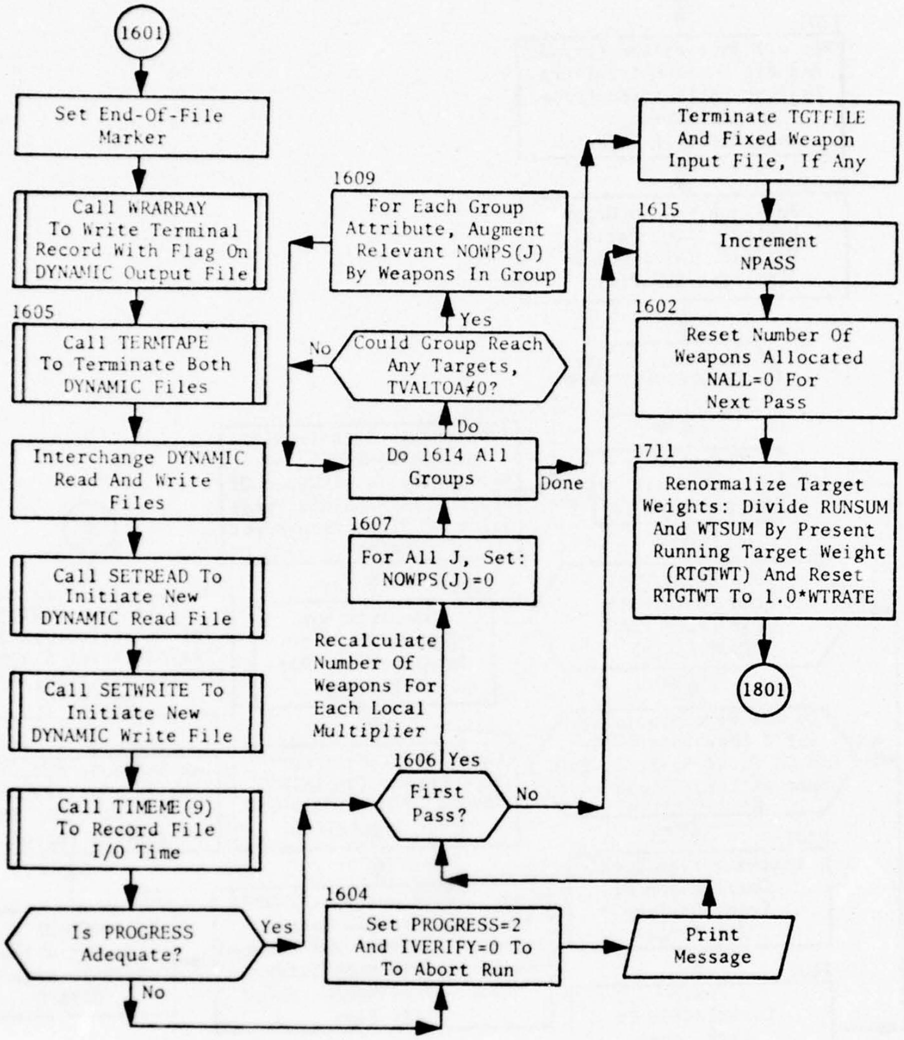


Figure 37. Part IV: (Part 5 of 5)

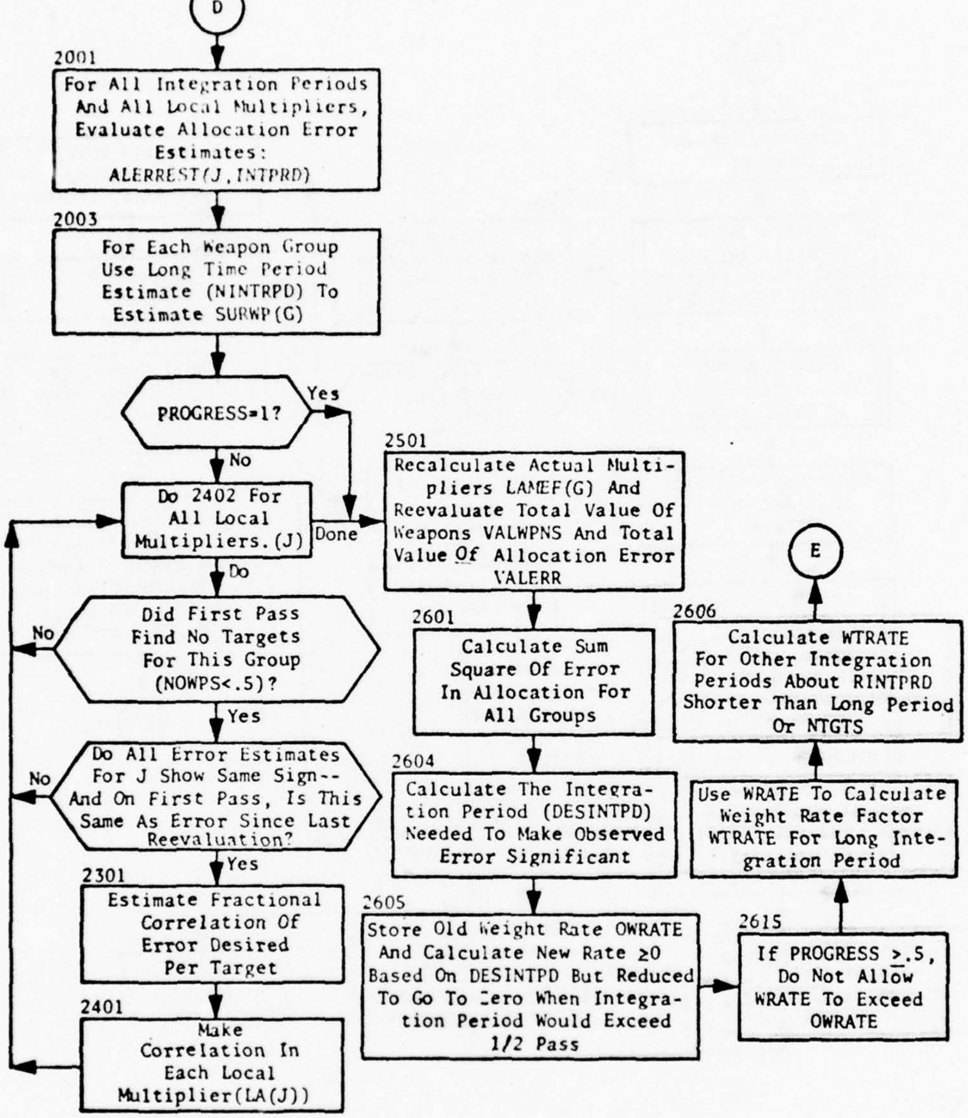
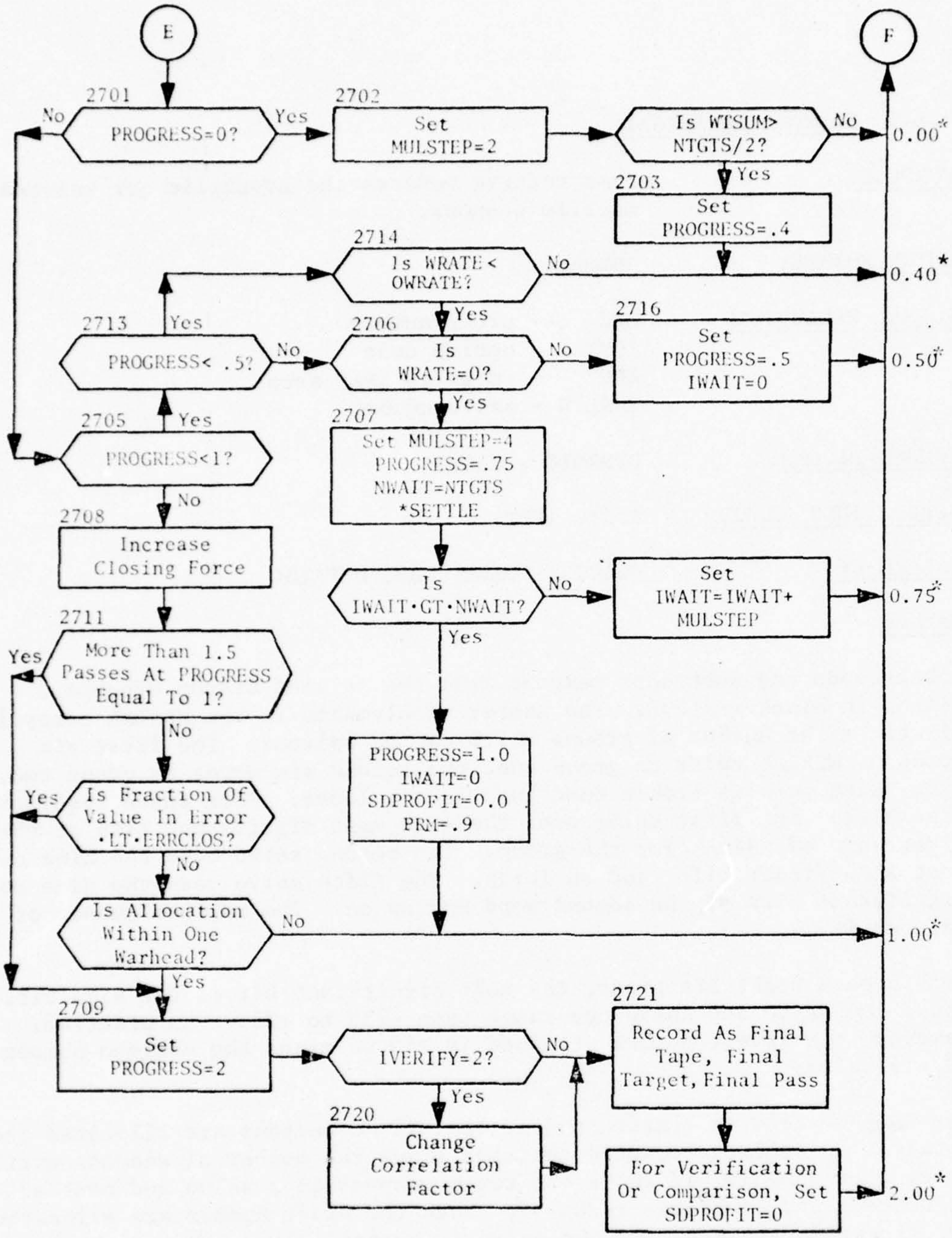


Figure 37. Part V: Multiplier Adjustment and Progress Evaluation (Part 1 of 2)



*Note: Starred numbers are values of parameter PROGRESS

Figure 37. Part V: (Part 2 of 2)

3.8.1 Subroutine ADDSAL

PURPOSE: This routine updates the stockpile for salvoed missile weapons.

ENTRY POINTS: ADDSAL

FORMAL PARAMETERS:
- MYG - group number
- IOPT - option code
- NUR - index to ISAL array
- ISALIN - salvo number

COMMON BLOCKS: DYNAMIC , SALVO

SUBROUTINES CALLED: IGET, IPUT

CALLED BY: MULCON, STALL, WAD, DEFALOC

Method:

ADDSAL adds and subtracts weapons from the salvoed weapon stockpile NSALAL in block /SALVO/. The number of elements in the NSALAL array is six times the number of groups which can be salvoed. The first six words of NSALAL refer to group one, the second six words to group two, etc. Each word is broken down into four salvoes. Each salvo requires nine bits. The first salvo uses the nine most significant bits of the first word of NSALAL for the group. The second salvo uses the nine next most significant bits, and so forth. The fifth salvo uses the nine most significant bits of the second word and so on. The maximum number of salvoes is 24.

Within each eight bit group, the most significant bit is the sign bit. Thus, the salvo stockpile can range from -255 to +255. In practice, however, the lowest number attained is -15 or minus the maximum number in any salvo.

The NSALAL array is constructed so that if no weapons are allocated from a salvo, the salvo stockpile contains minus the number of weapons available. For example, if there are seven weapons in a salvo and none are allocated, the stockpile reads -7. When the exact number are allocated, the stockpile is zero. If the salvo is overallocated, the stockpile is the size of the overallocation. For example, if a salvo contains five weapons and eight are allocated, the stockple will read +3.

The formal parameter IOPT is an addition/deletion indicator. If IOPT = 3, weapons are to be added. If IOPT = 4, weapons are to be deleted. This code is the same as for the option index WADOP used by STALL and WAD.

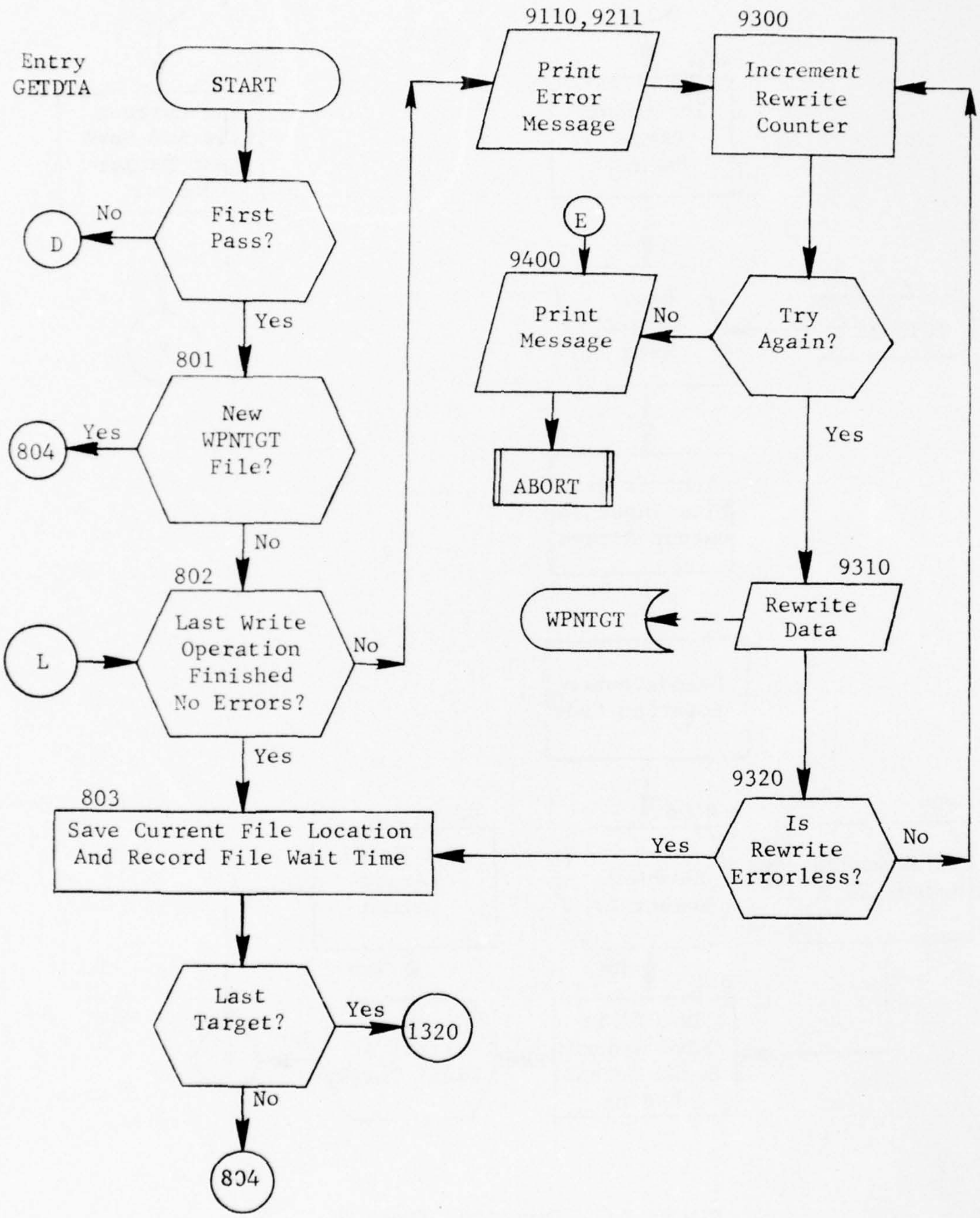


Figure 41. Part II: Input Filehandling
(Part 1 of 2)

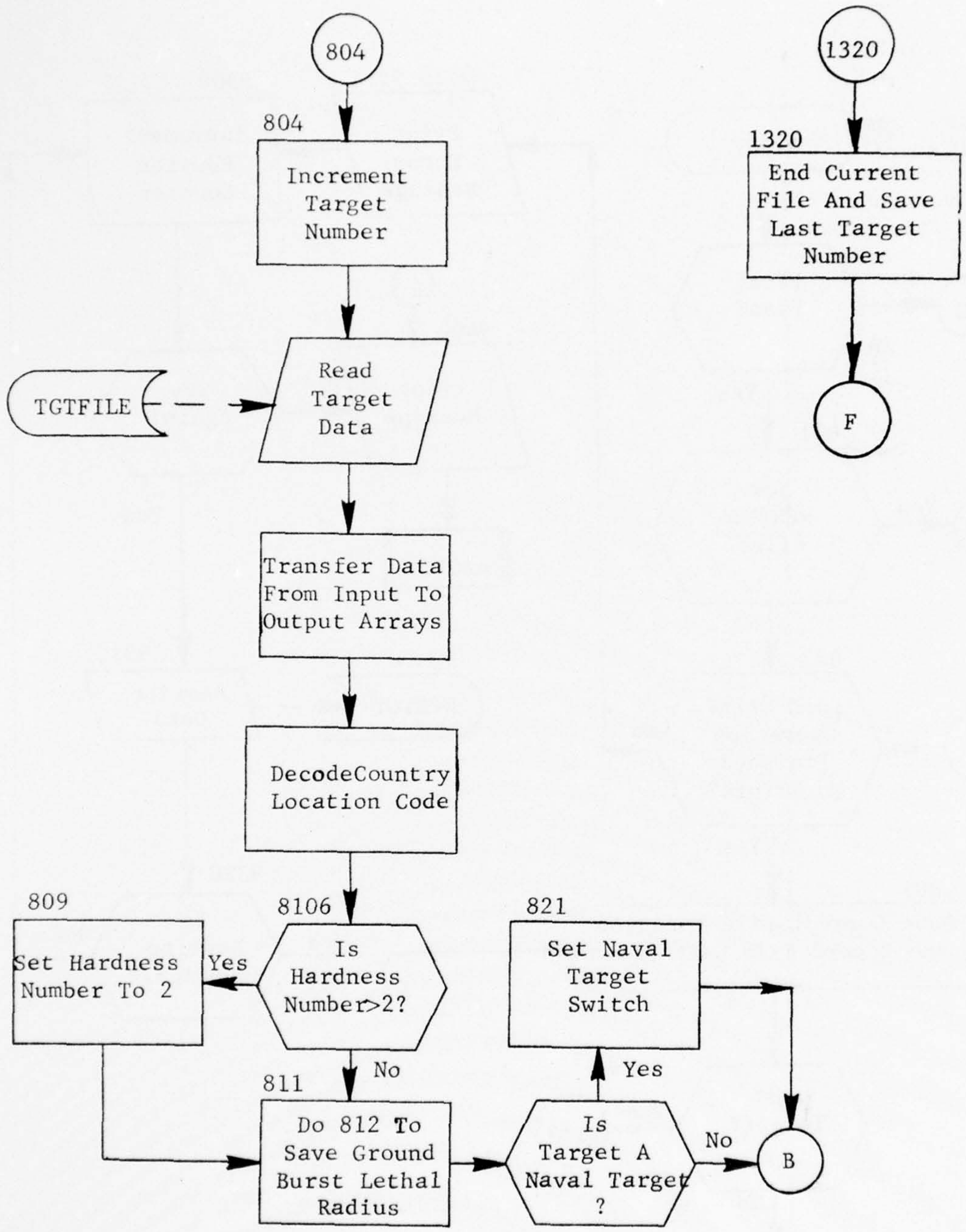


Figure 41. Part II: (Part 2 of 2)

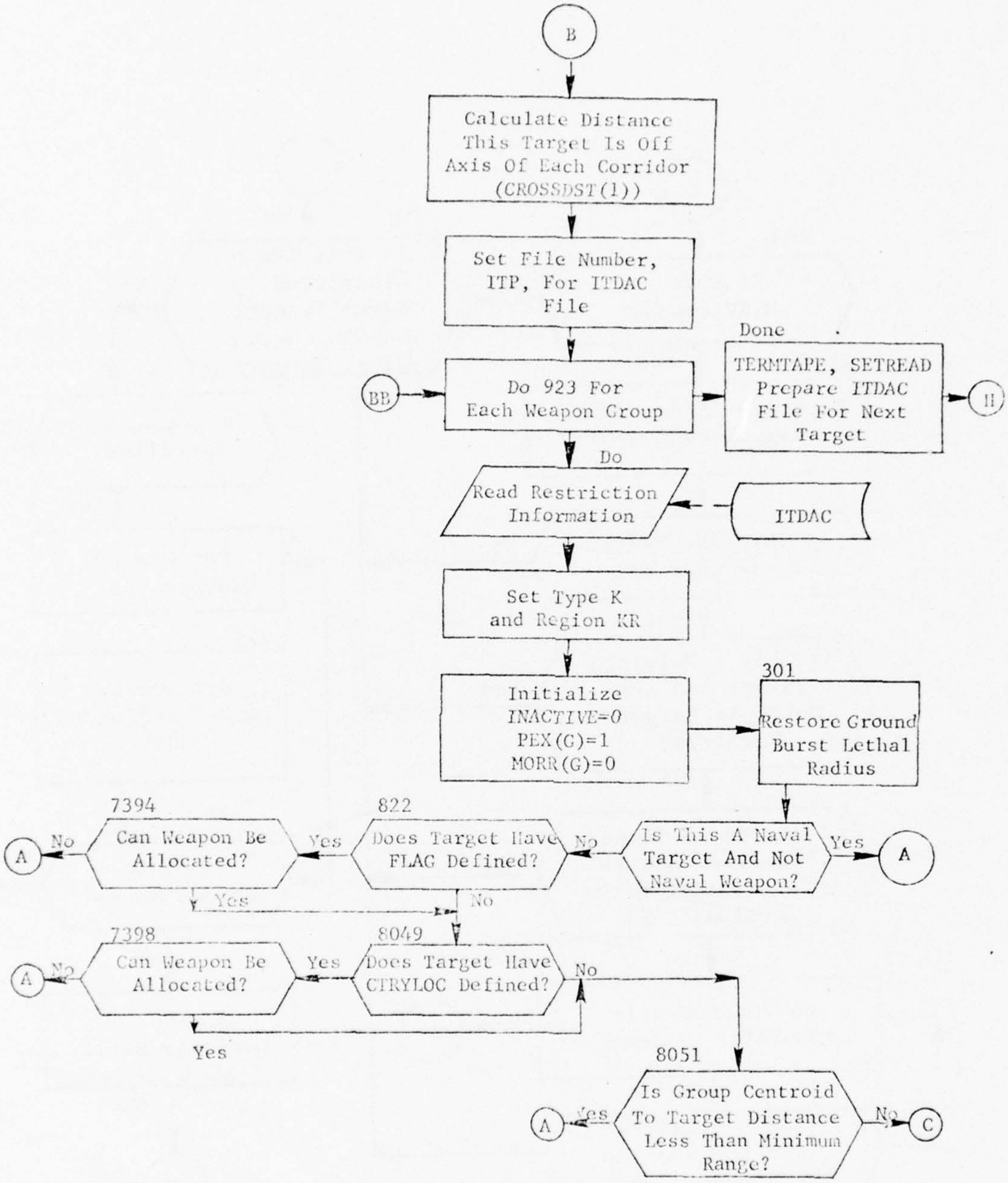


Figure 41. Part III: Weapon-Target Calculations (Part 1 of 7)

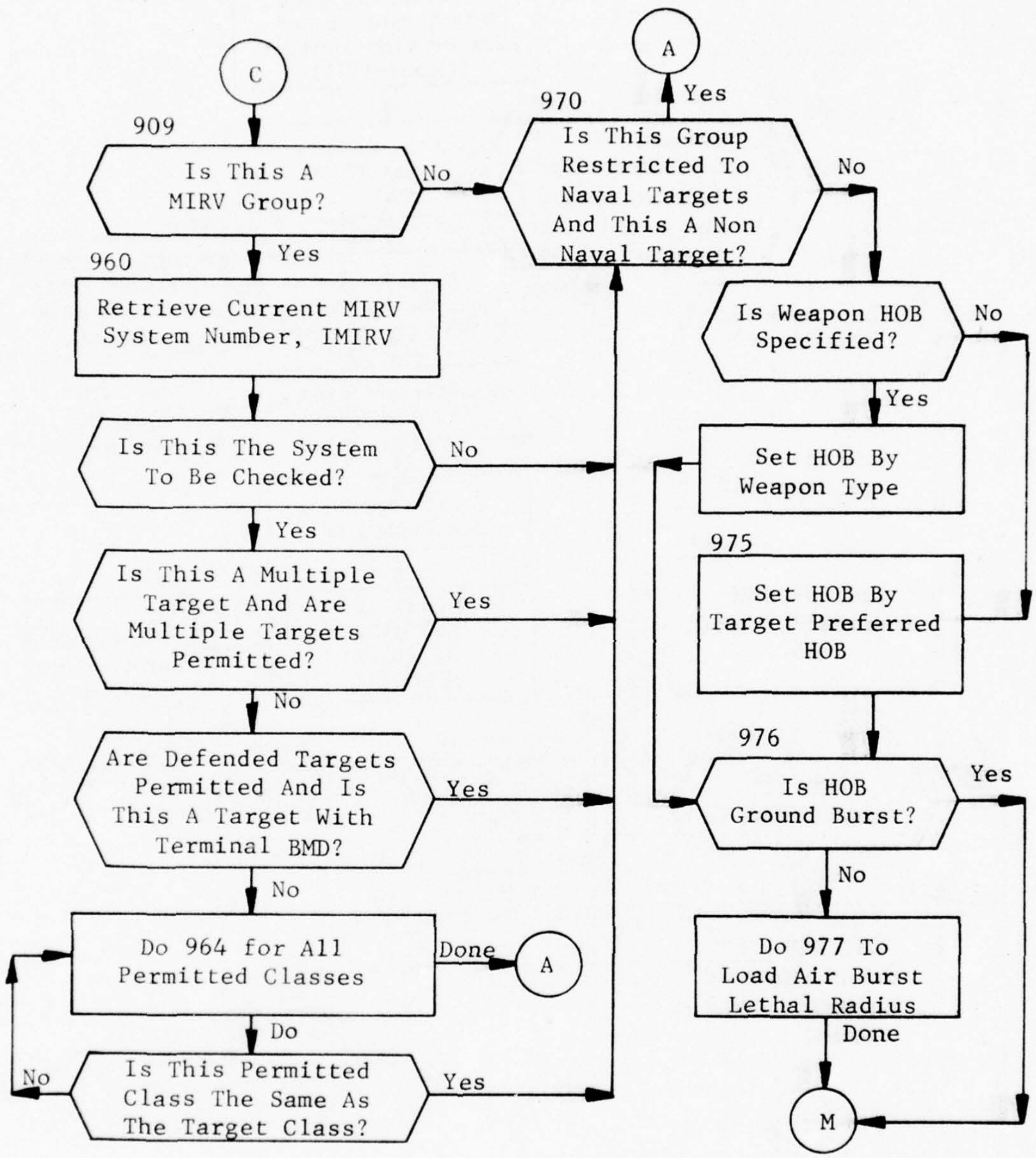


Figure 41. Part III: (Part 2 of 7)

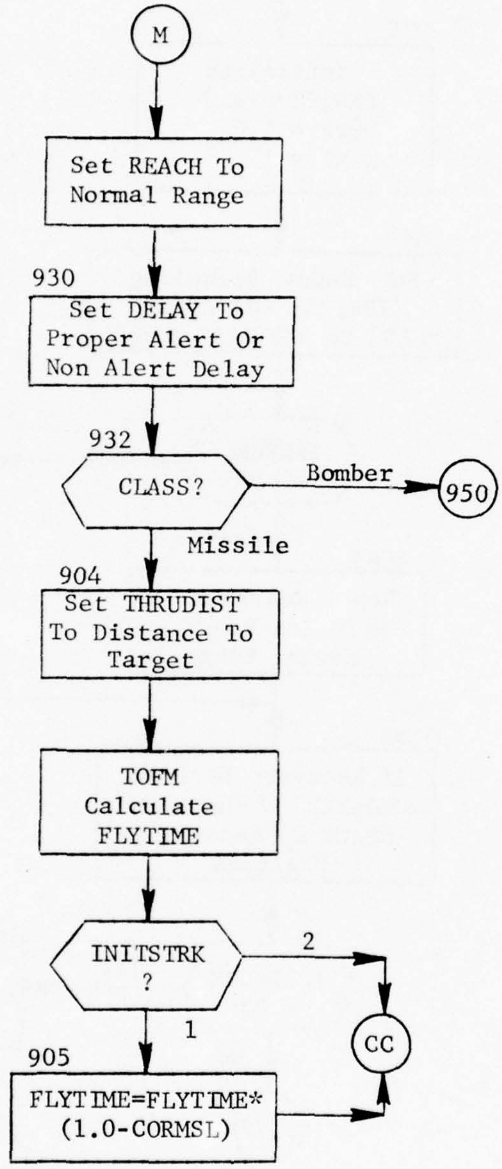


Figure 41. Part III: (Part 3 of 7)

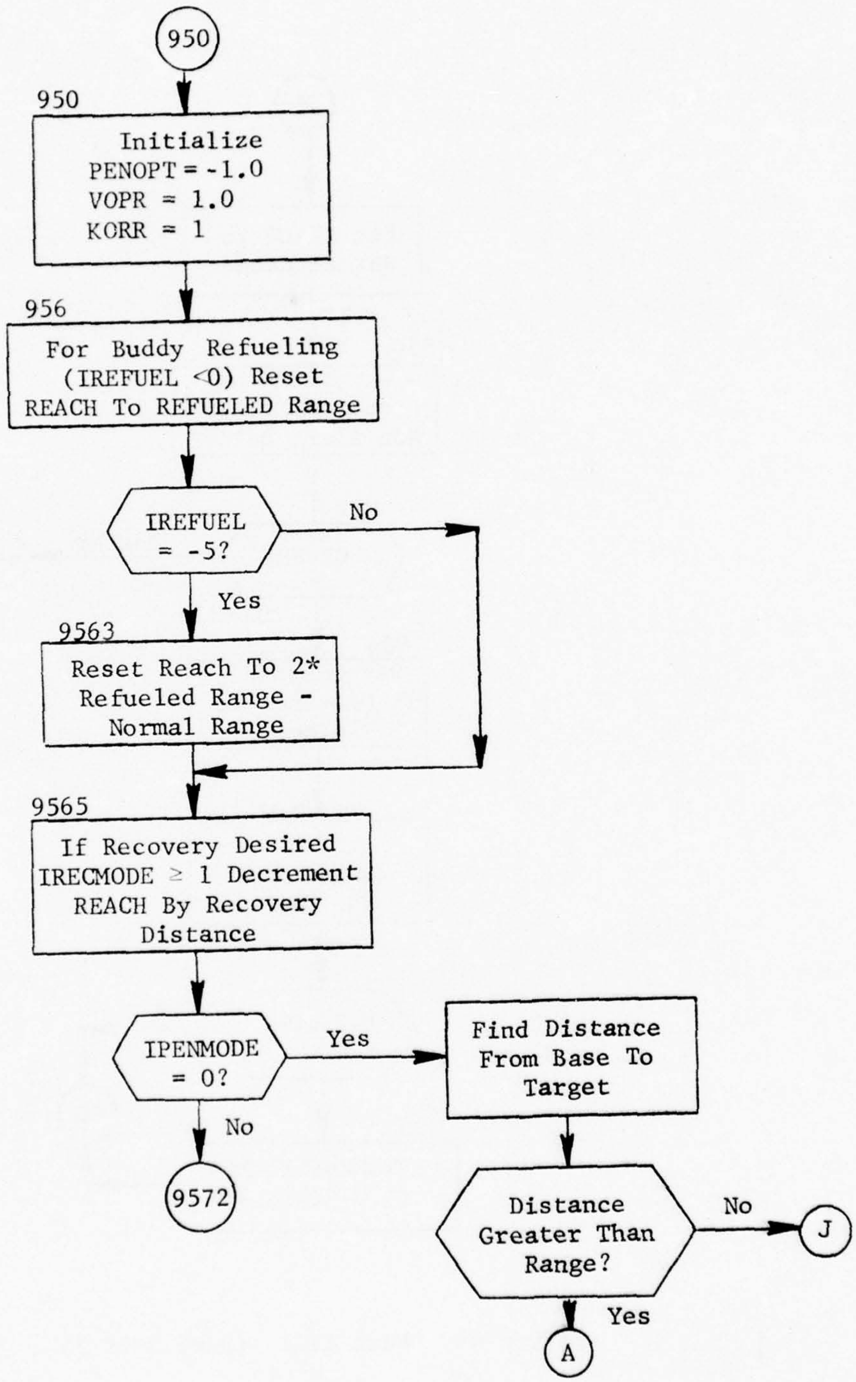


Figure 41. Part III: (Part 4 of 7)

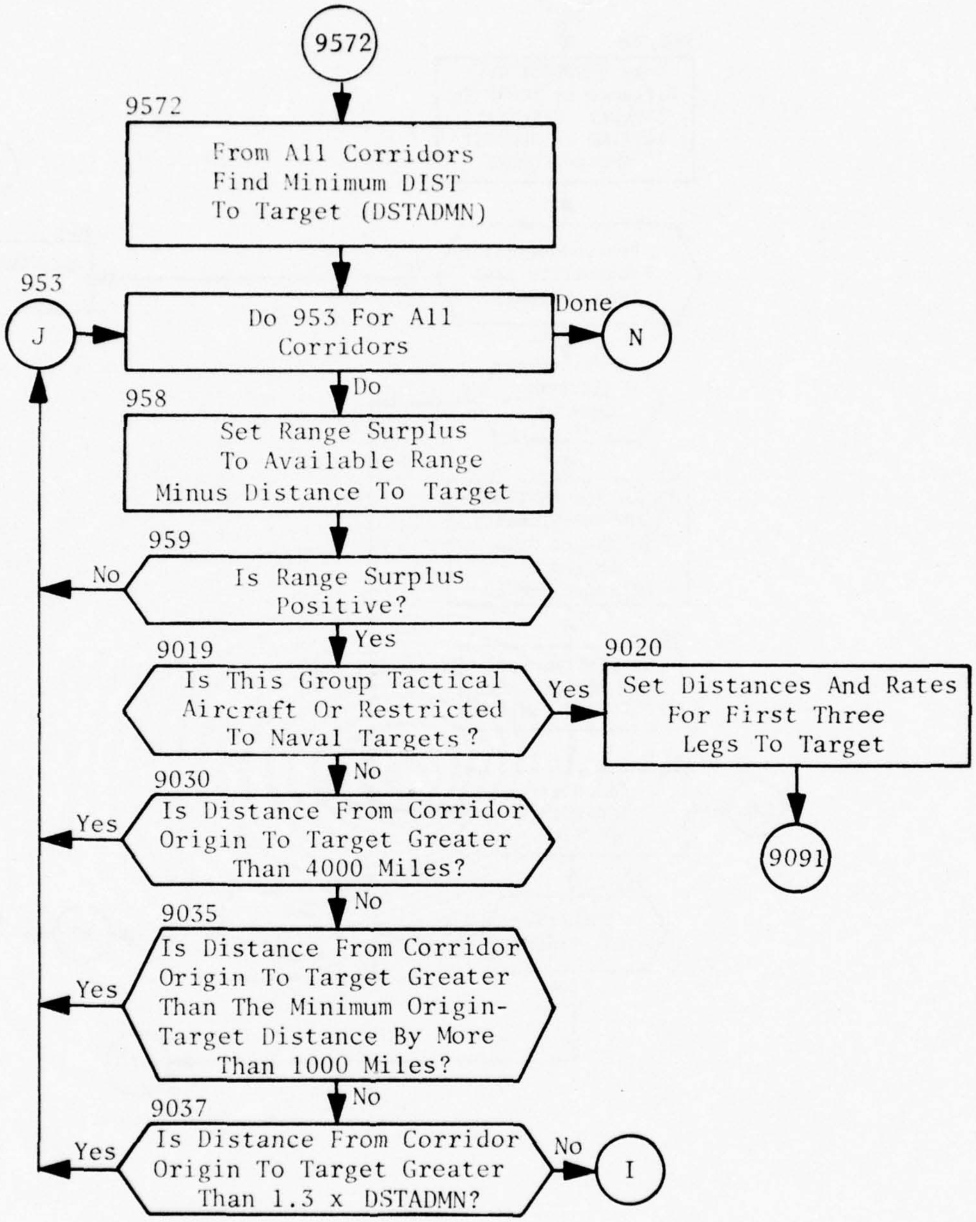


Figure 41. Part III: (Part 5 of 7)

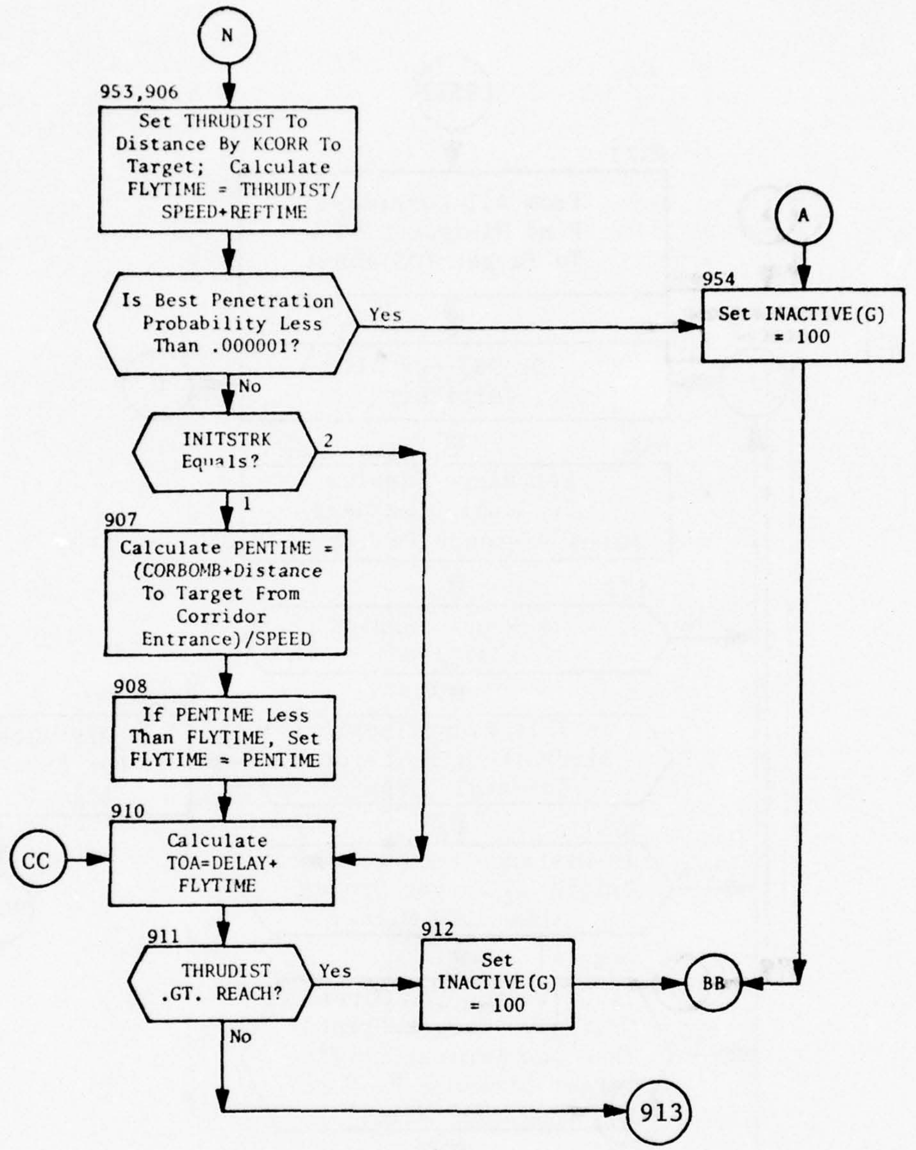


Figure 41. Part III: (Part 6 of 7)

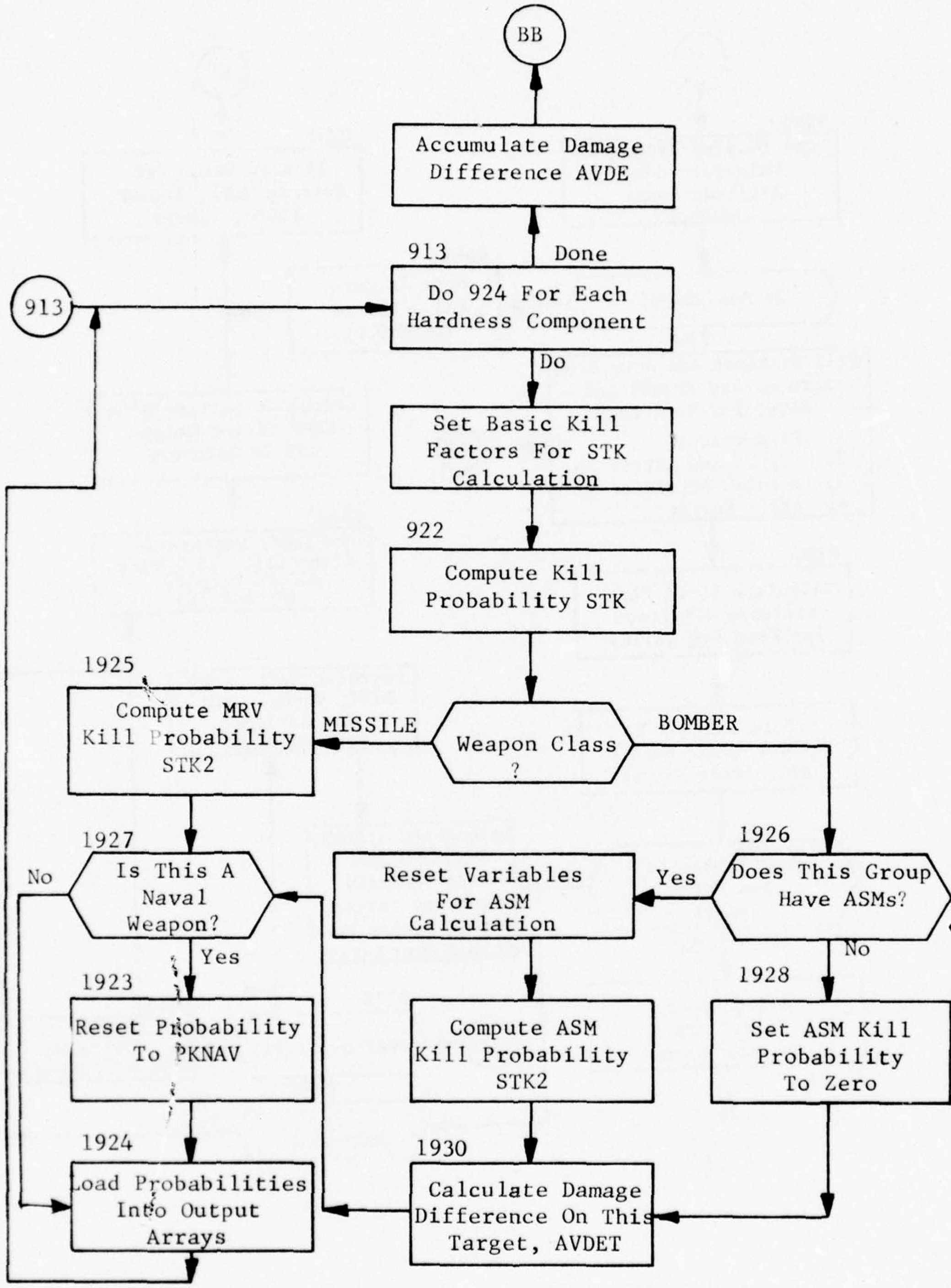


Figure 41. Part III: (Part 7 of 7)

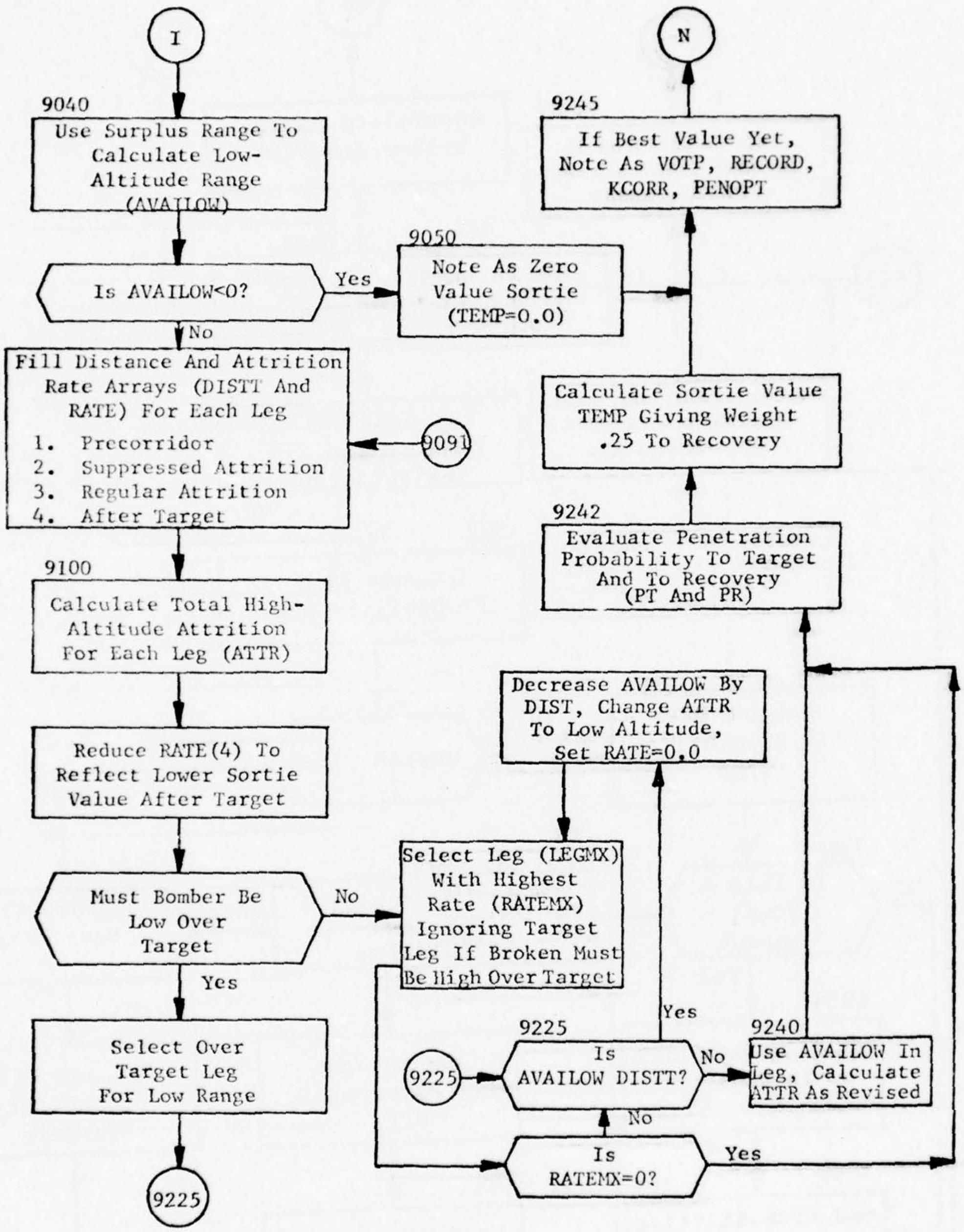


Figure 41. Part IV: Bomber Penetrability

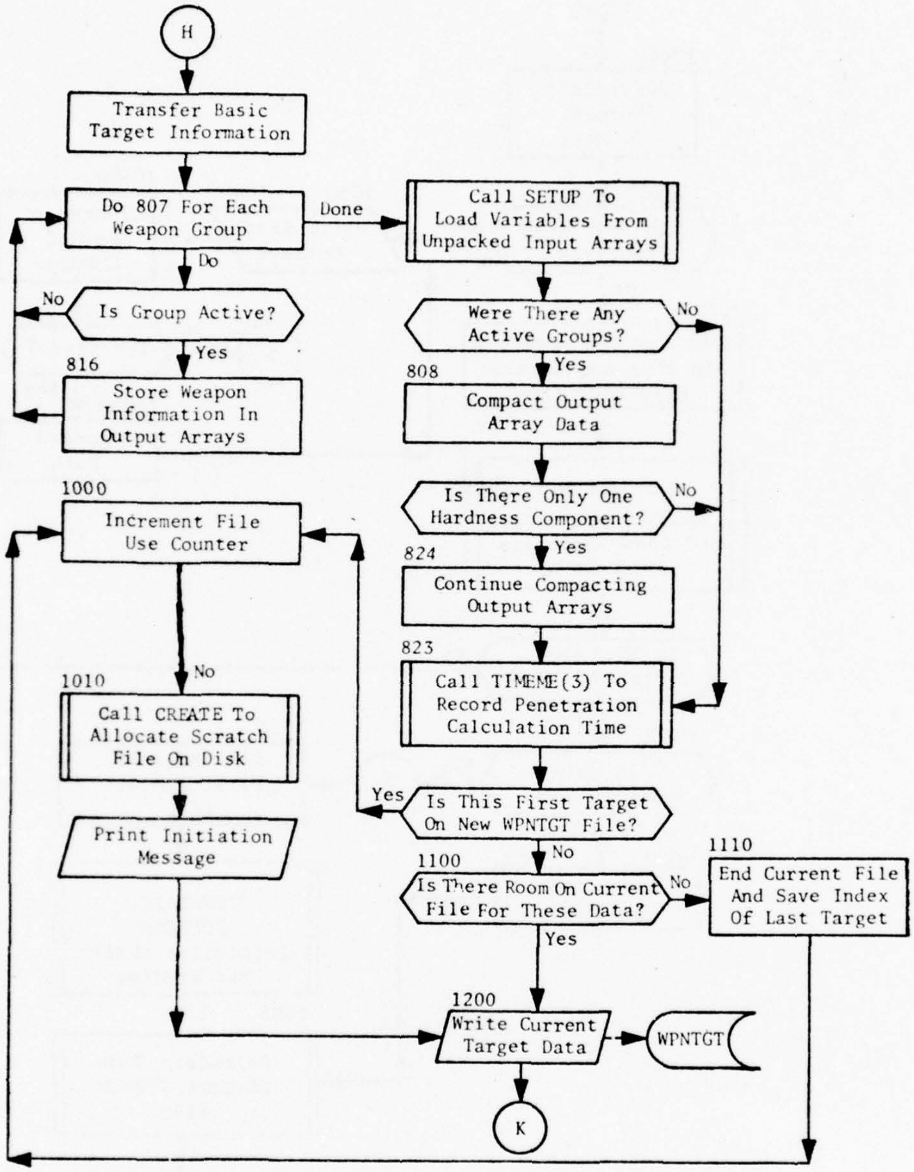


Figure 41. Part V: Output Array Processing

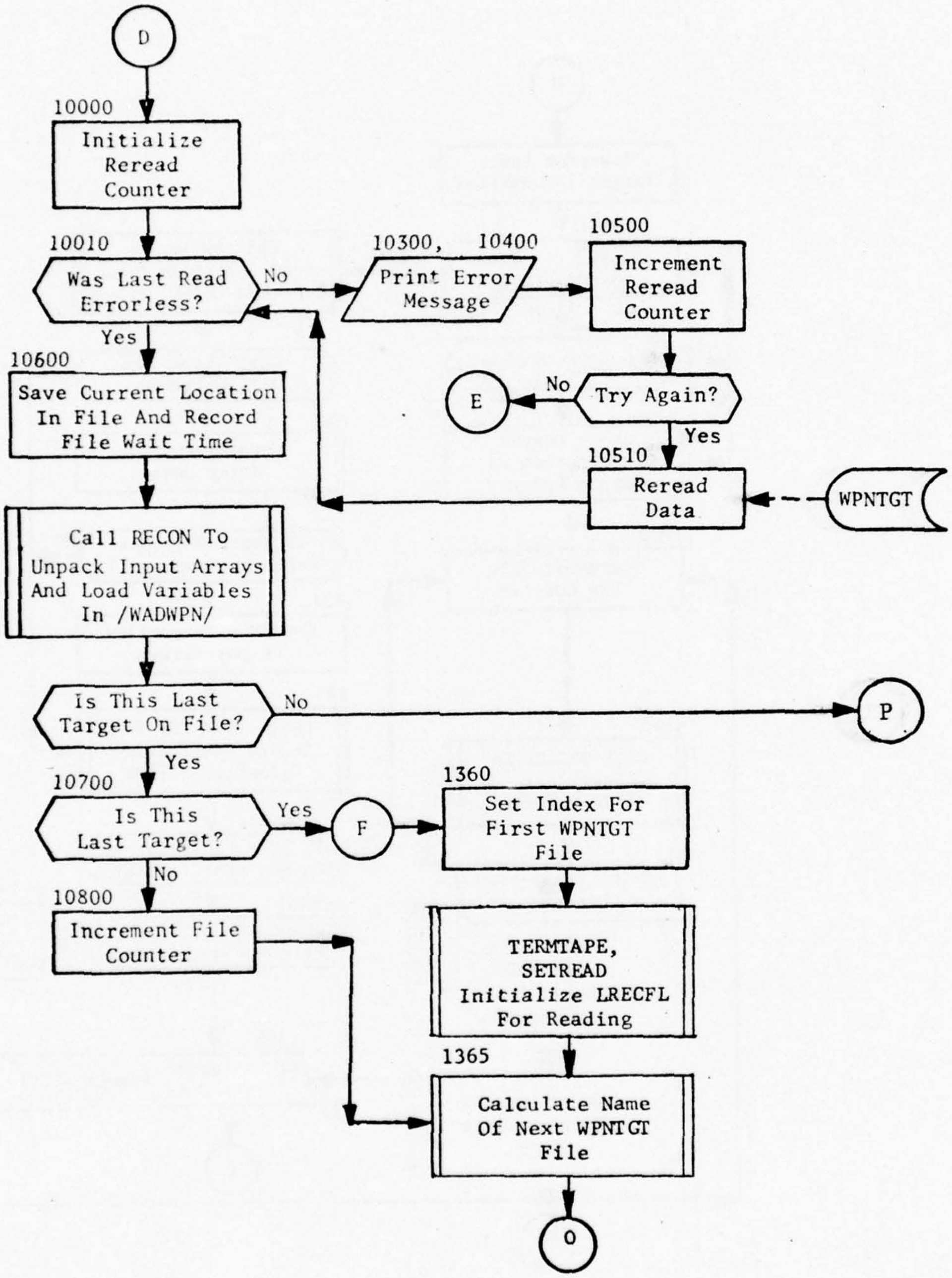


Figure 41. Part VI: Second Pass Processing (Part 1 of 2)

3.8.5 Function LAMGET

PURPOSE: This real valued function calculates the Lagrange multiplier for salvoed missile groups.

ENTRY POINTS: LAMGET

FORMAL PARAMETERS:
LAM - Initial multiplier
P - Salvo balance variable
ISAL - Salvo number

COMMON BLOCKS: None

SUBROUTINES CALLED: None

CALLED BY: DEFALOC, SALVAL(entry INITSAL)

Method:

Note that this function is a real valued function. Its correct usage requires that a REAL LAMGET specification be present in the calling program.

The formal parameters specify the original or first salvo multiplier (LAM), the salvo balance variable maintained by PUPDT (P), and the salvo number (ISAL).

The returned value LAMGET is computed as follows:

```
LAMGET = LAM - (P * (ISAL - 1)) * LAM
```

Function LAMGET is illustrated in figure 42.

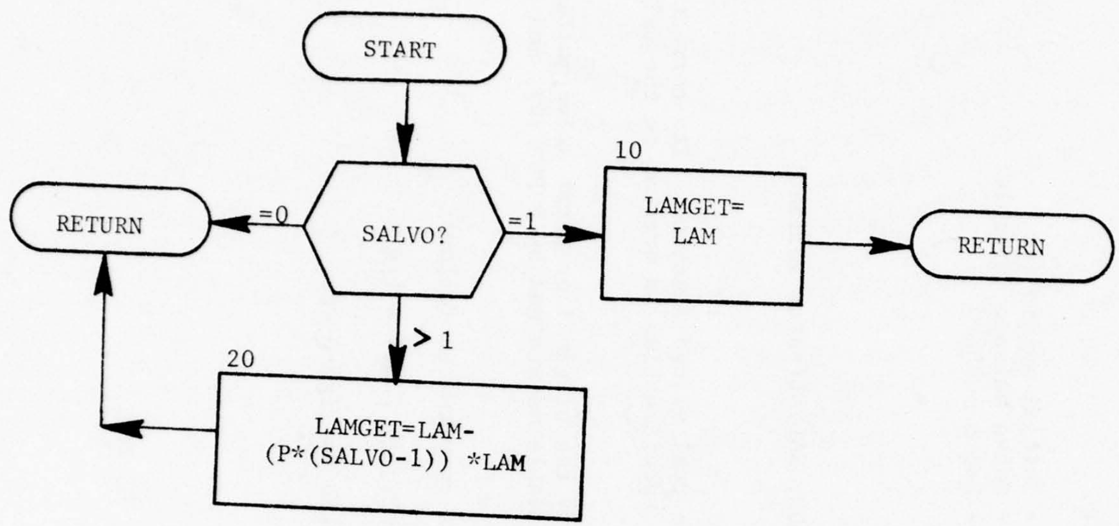


Figure 42. Function LAMGET

PRNTCON is called by MULCON before proceeding to process each new target. PRNTCON first reinitializes all print control flags (IDO(INDEXPR) for regular prints and IFTPRNT(INDEXPR - 100) for FILEHANDLER prints) to a nonprint state (IDO = 1, IFTPRNT = 0). It then examines the list of print requests to see if any are operative for this target on this pass. For each operative print the flags are set to print (IDO = 3, IFTPRNT = KTGTFREQ).

This arrangement makes it possible to request the same print at different targets or passes with separate independent print requests. In the case of file prints, if more than one request on the same LTN are simultaneously operative, the last request will control the number of words prin printed. If regular prints are requested with KTGTFREQ greater than 1, the first print will not occur at JTGT but at KTGTFREQ - 1 targets later, and thereafter the print will occur every (KTGTFREQ)th target.

Figure 45 illustrates subroutine PRNTCON.

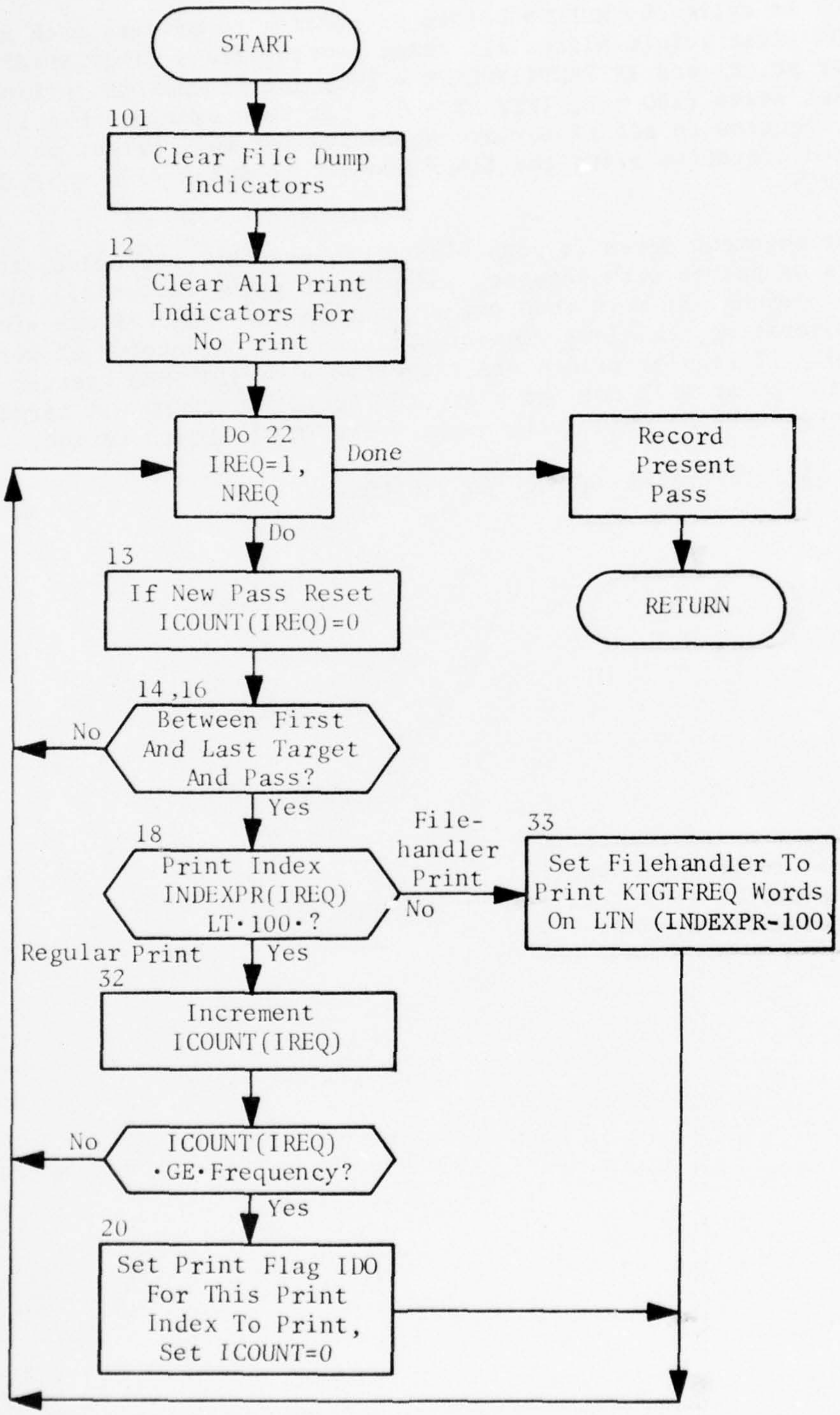


Figure 45. Subroutine PRNTCON

3.8.9 Subroutine RECON

PURPOSE:	This routine reconstructs the weapon target interaction data from the compressed format used on the WPNTGT files. Entry SETUP is used in the first pass when no decompression is necessary.
ENTRY POINTS:	RECON, SETUP
FORMAL PARAMETERS:	None
COMMON BLOCKS:	CONTROL, DEFENSE, DYNAMIC, PAYLOAD, PAYOFF, PKNAVAL, SALVO, SMAT, WADWPN, WPNGRP, WPNREG, WPNTYPE
SUBROUTINES CALLED:	SETPAY, TABLEMUP, TIMEME, VALTAR
CALLED BY:	GETDTA

Method:

The first part of this routine is used only after pass one. Before writing the weapon target interaction data on the WPNTGT files, subroutine GETDTA compresses the data by removing spaces reserved for inactive groups. If the target has only one hardness component, space reserved in the STK and STK2 arrays in common /WADWPN/ for the second component is also removed. The first part of RECON, therefore, merely decompresses the data. The VTOA and XMUP arrays of common /WADWPN/ are used for temporary storage for the STK and STK2 data, since these later arrays may be overwritten by the data input from the WPNTGT files.

Entry SETUP is used on all passes (but not explicitly on passes after the first). Subroutine SETPAY is called to select use of gravity bombs or ASMs from the bomber groups. If ASMs are selected probability STK2X is used in the calculation of the survival probability STK and the MUP array. The MUP, XMUP, SSIG, and RISK arrays are constructed according to the required formulae.

The processing in statements 943 and 960 is used to load the correct SMAT array. Through the use of the user-input parameter FACMIRV, the SMAT array may differ for MIRV and nonMIRV weapons.

In statement 1925 the data base attribute PKNAV is used for the single shot kill probability for weapons which have nonzero values for the attribute.

Subroutine RECON is illustrated in figure 46.

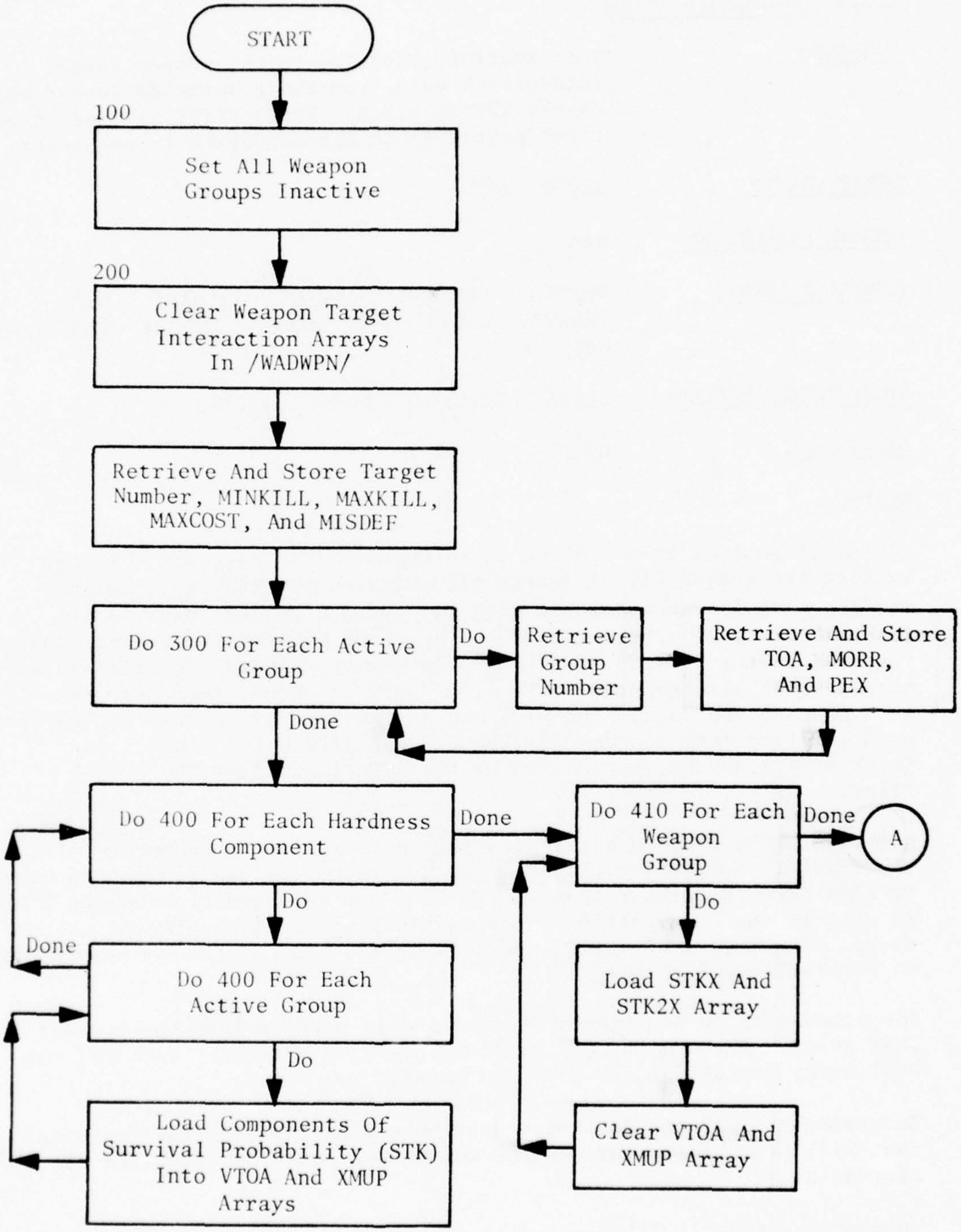


Figure 46. Subroutine RECON
Part I: Entry RECON

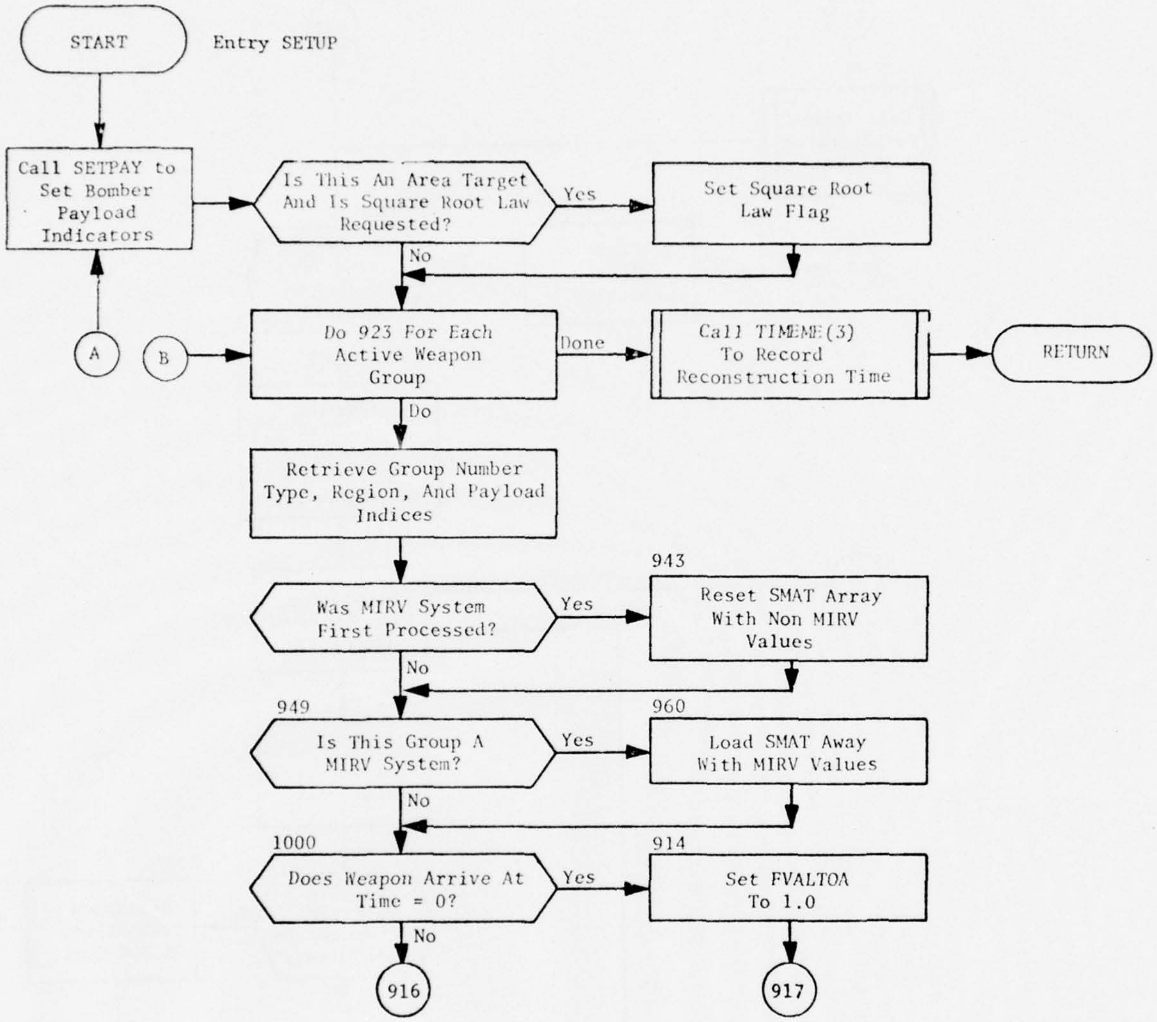


Figure 46. Part II: Entry SETUP
(Part 1 of 2)

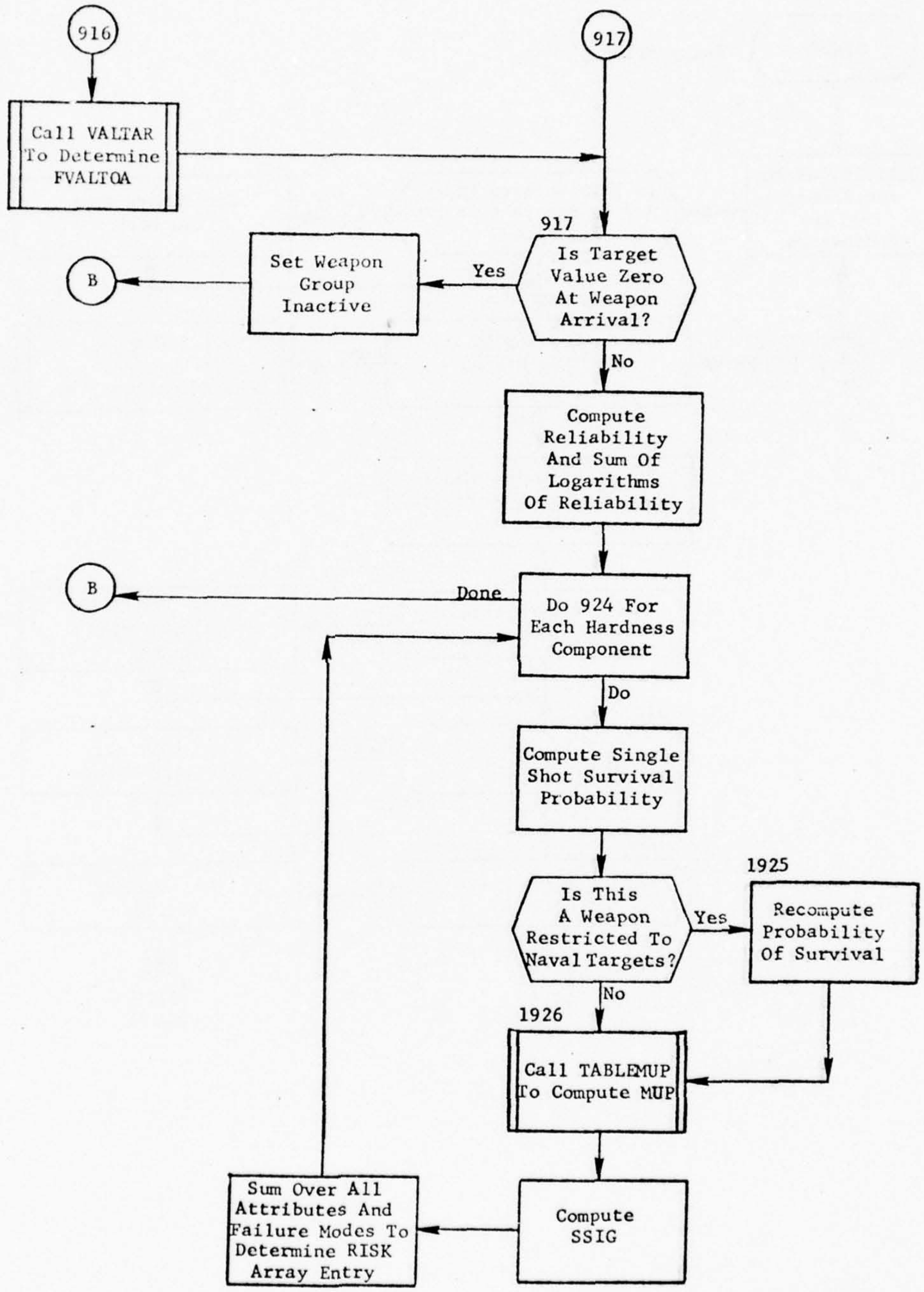


Figure 46. (Part 2 of 2)

Note that if a salvo is unavailable because of limit on a call to NEWSAL, the routine attempts to find the closest lower available salvo. If none can be found, then MYSAL is set negative to flag that no salvo is available for this group.

Entry NEWSAL is illustrated in part 4 of figure 47.

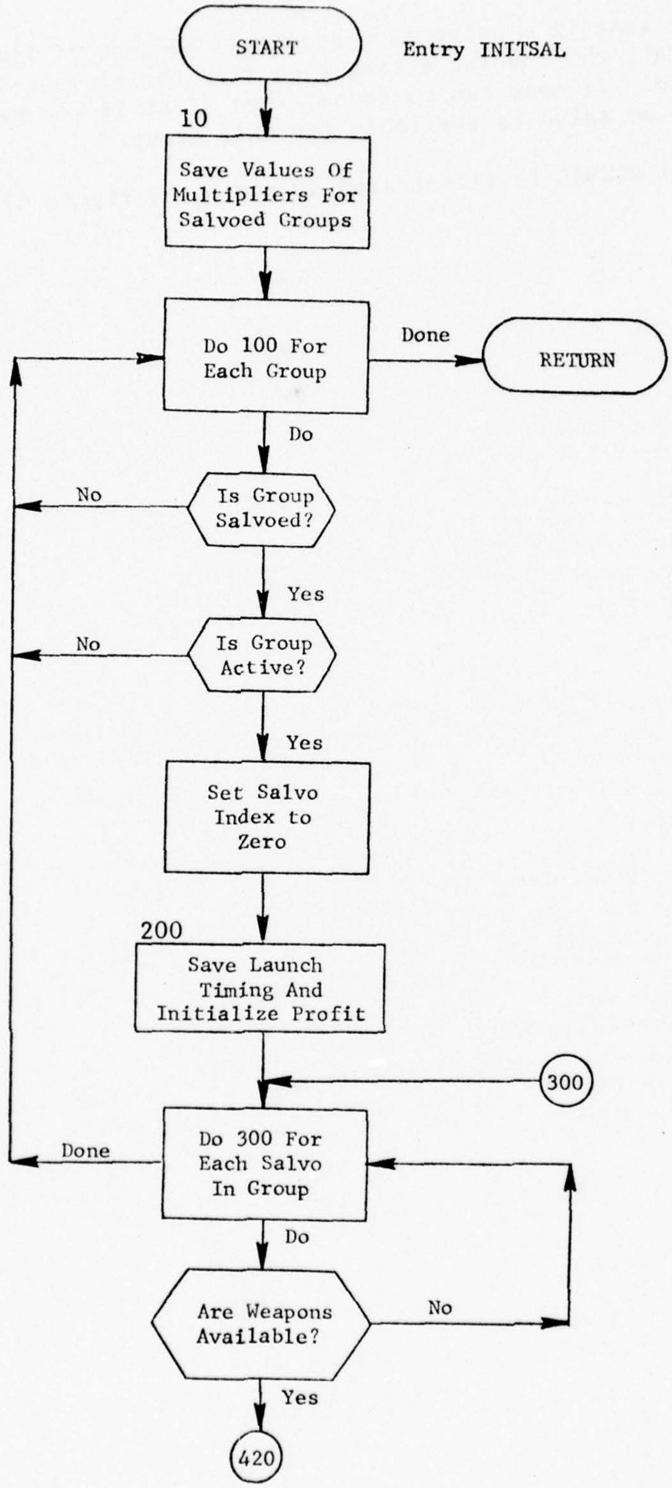


Figure 47. Subroutine SALVAL
(Part 1 of 4: Entry INITSAL)

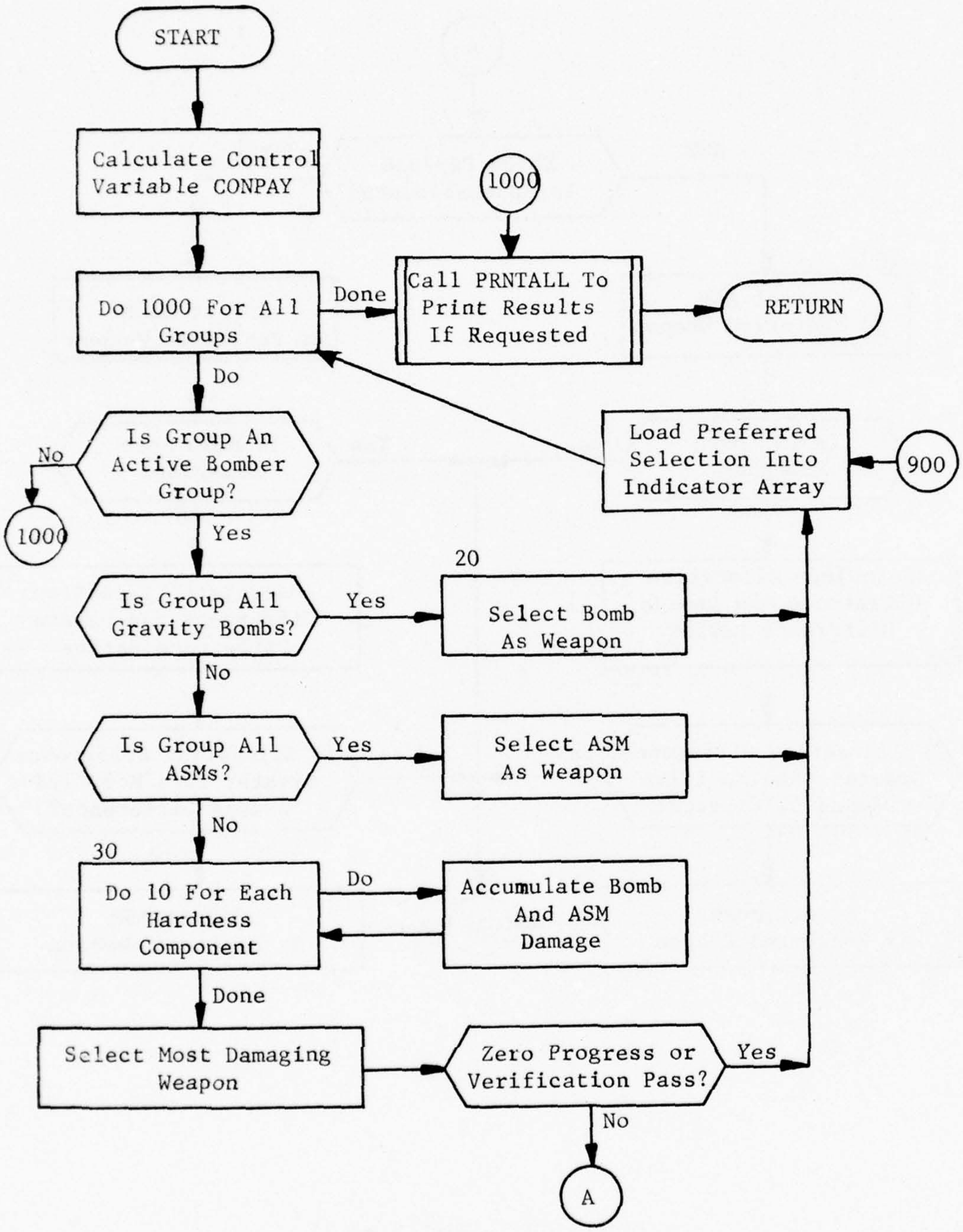


Figure 49. Subroutine SETPAY (Part 1 of 2)

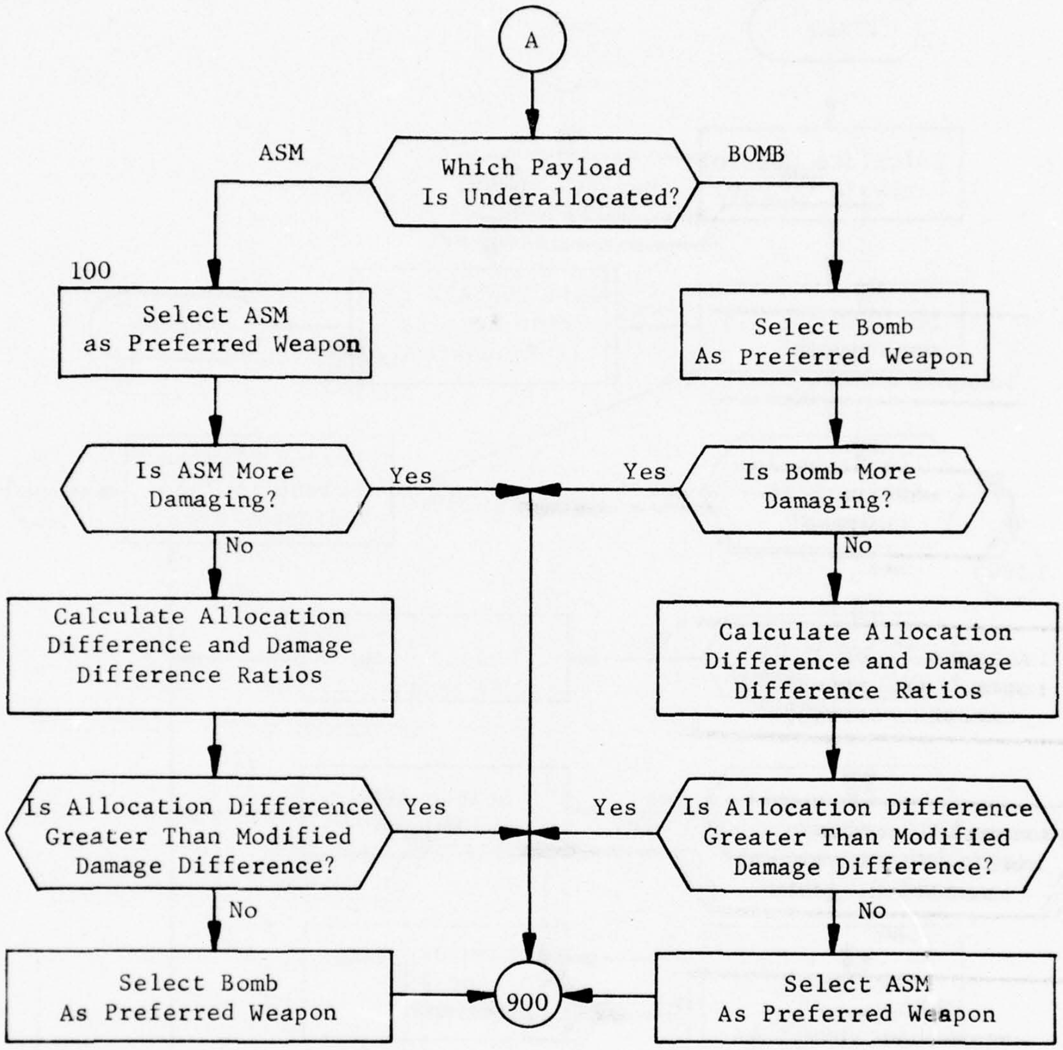


Figure 49. (Part 2 of 2)

were added. These data are stored in the array SIGP(G,J,N). Using these data, the effective augmentation of SIG to calculate VTP(G) for each weapon group G can be accomplished simply by adding SIGP for the appropriate column, and the augmentation of MU is accomplished simply by adding MUP for each weapon G.

Table 15-A may help to visualize how these data are used. Each potential group G is tagged with the index ITOA(G) of the time of arrival column it would occupy if it were added. In addition it is also noted whether the weapon would generate a new column (IADDTOA=1) or share the column with the weapons already there (IADDTOA=0). The situation illustrated here in table 15 corresponds to the same one illustrated in table 14 Notice that for each weapon group G the array SIGP(G,J,NI) contains a significant (usually nonzero) data for NI = ITOA(G) - IADDTOA. The extra term in the column NI = ITOA(G)-1 for rows 1 and 5, where the weapon group would require a new column (IADDTOA=1), is to provide a term required for the new columns if this weapon were added. Since addition of this weapon would move all columns (including its own) one position to the right, the resulting term would be in the proper position after moving, even though it is in an incorrect column at present.

Just as the information in table 15-A is used to provide values of SIG for computing VTP(G), the array SIGD in table 15-B is used to provide values of SIG for the computation of VTD(NW). This table contains an entry for each weapon currently on the target. The array IG(NW) in this case indicates that three weapons have been assigned, first from group 2, then group 3, finally another from group 3. The role of SIGD exactly parallels SIGP; that is, to obtain the potential value of SIG if a weapon were deleted SIGD is added to SIG in each column. Since SIGD is negative, this has the effect of cancelling out the cross terms for the weapon that would be removed. Of course, if removal of the weapon would reduce the number of weapons in a time-of-arrival column to 0, the following columns would be spaced back to avoid unnecessary columns.

In summary, table 14 contains the scratch pad data used to calculate the actual payoff. Table 15-A contains the corrections SIGP for SIG needed to calculate the corresponding weapon-added estimate VTP(G) for each weapon group G. Table 15-B contains the corrections SIGD for SIG needed to calculate the weapon-deleted estimate VTD(NW) for each weapon NW now on the target.

These arrays SIGP and SIGD are kept continuously up-to-date as weapons are added and deleted. For example, as illustrated in table 15, the last weapon added was from group 3. Thus the last set of cross terms computed would have been the cross terms between group 3 and every other group. These cross terms are shown in table 15-A in the array DSIG(G,J). when the last weapon from group 3 was added, these terms were computed, and for each weapon G they were added into the array SIGP in all time of

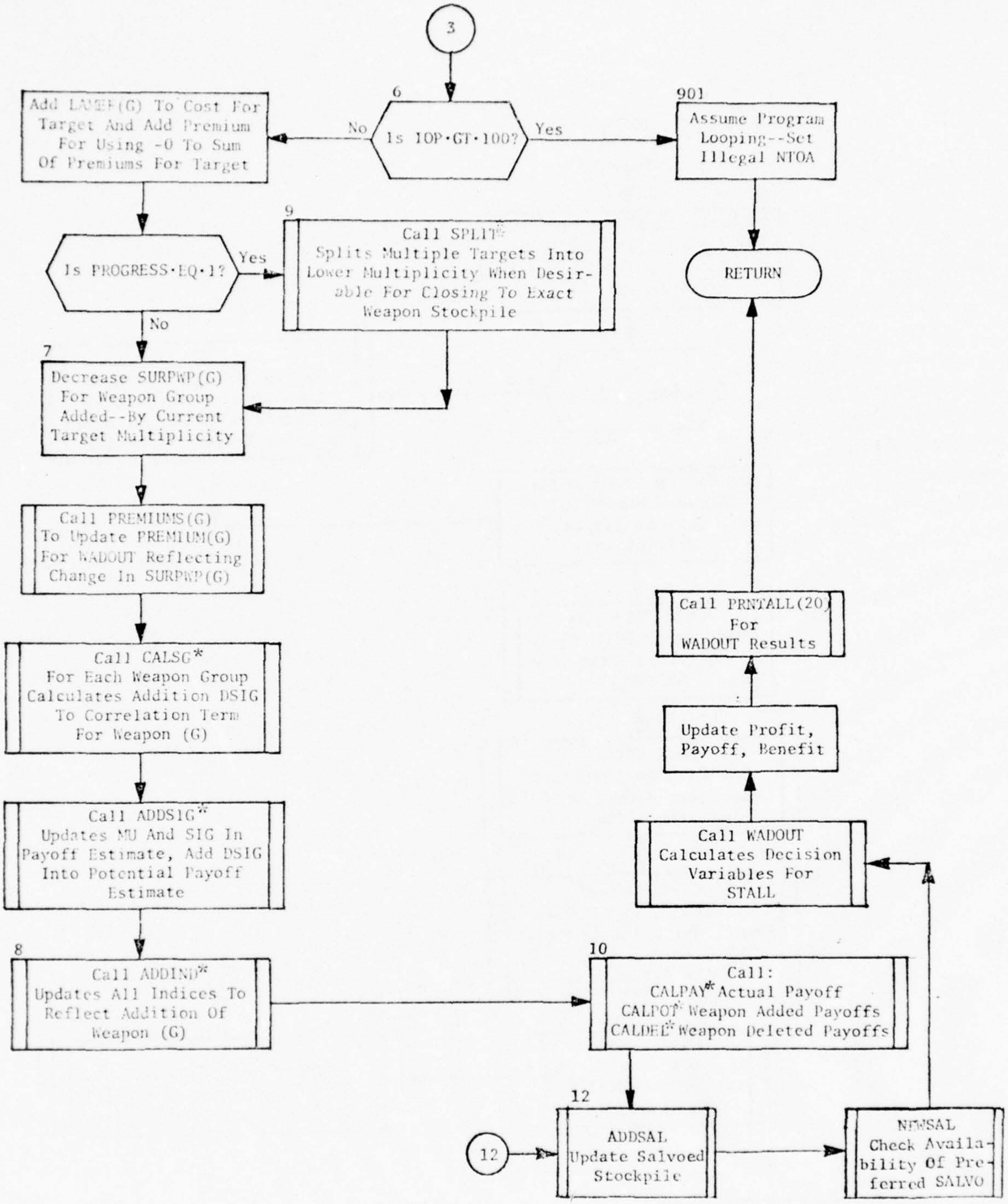
Table 15. Illustrating Quantities Calculated for Potential Weapon Added and Deleted Payoffs

A. - Data on All Potential Weapons

G	VTP (G)	ITOA (G)	IADDTOA (G)	J	DSIG, 3	SIGP(G, J, NI) NI=1	NI=2	NI=3	NI=4
1		2	0	1	.021	0.0	.042	.063	
				2	.103	0.0	.206	.659	
2		3	0	1	.052	0.0	0.0	.45	
				2	.660	0.0	0.0	4.23	
3		2	0	1	.274	0.0	.548	1.653	
				2	.780	0.0	1.560	1.880	
4		3	0	1	.005	0.0	0.0	.010	
				2	.020	0.0	0.0	.040	
5		4	1	1	.003	0.0	0.0	.007	
				2	.009	0.0	0.0	.018	

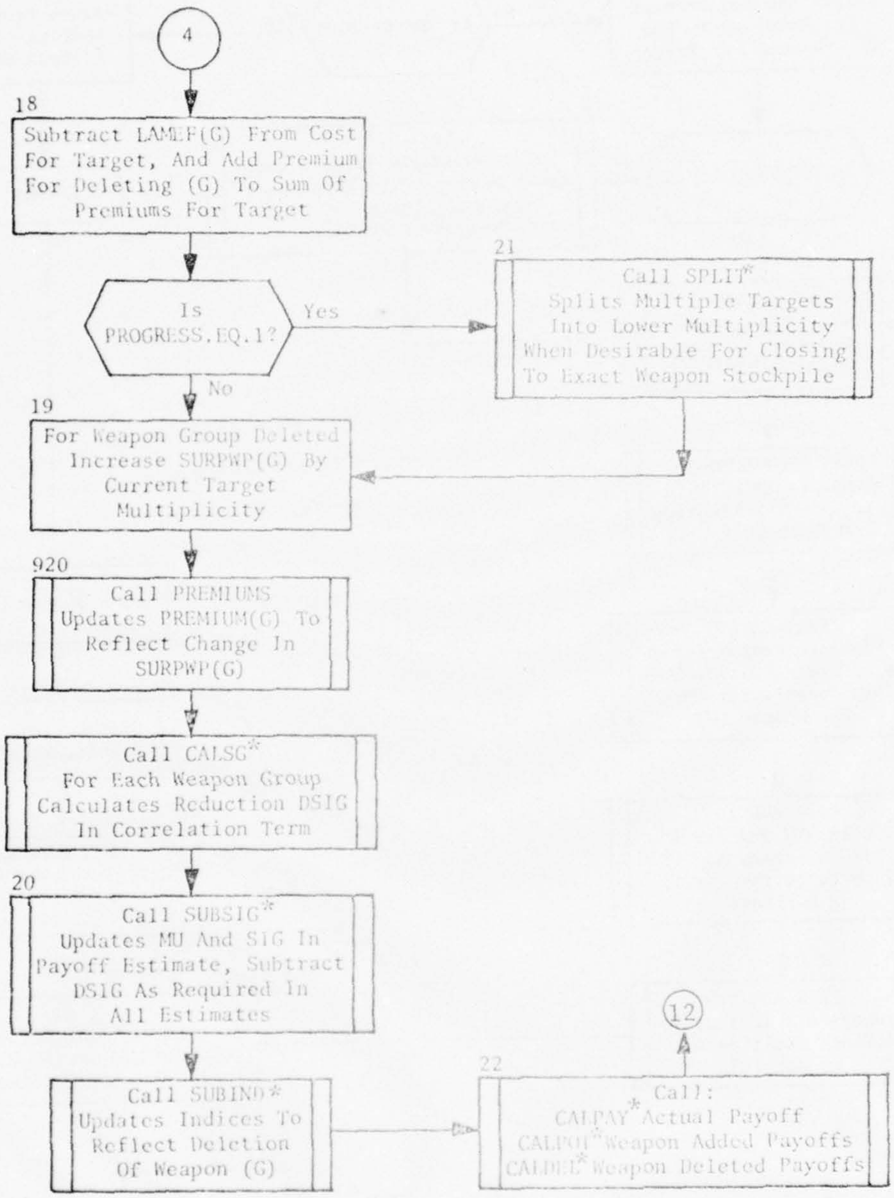
B. - Data on Weapons Now on Target
(As Candidates for Possible Deletion)

NW	IG(NW)	VTD(NW)	J		SIGD(NW, J, NI) NI=1	NI=2	NI=3	NI=4
1	2		1		0.0	0.0	- .105	
			2		0.0	0.0	-1.32	
2	3		1		0.0	- .274	- .052	
			2		0.0	-1.560	- .660	
3	3		1		0.0	- .274	- .052	
			2		0.0	-1.560	- .660	



*Local Subroutines

Figure 53. Part III: Add Weapon Control



*Local subroutine

Figure 53. Part IV: Delete Weapon Control

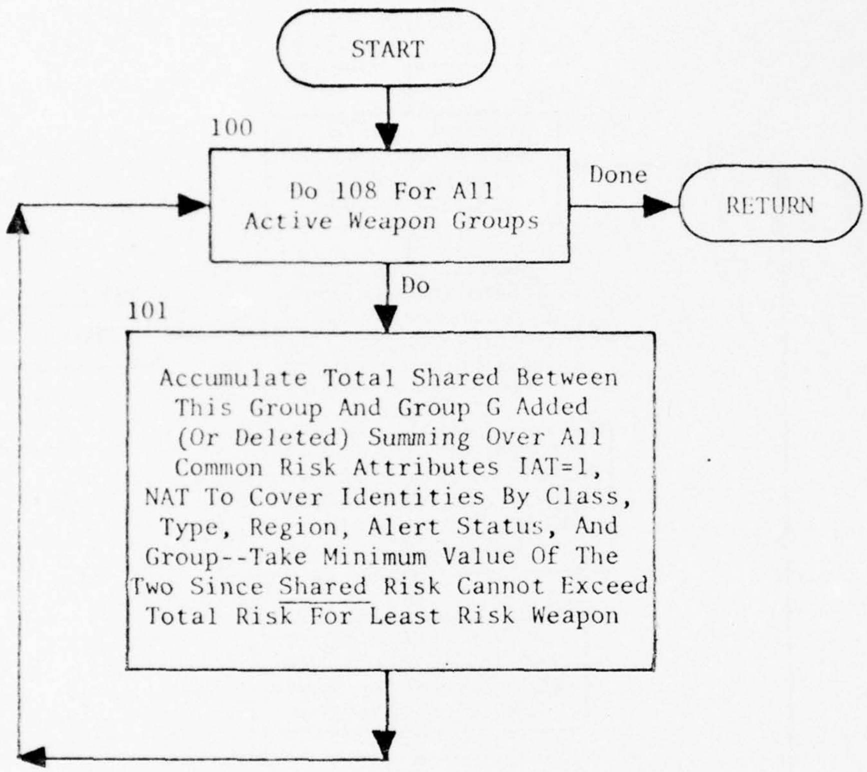
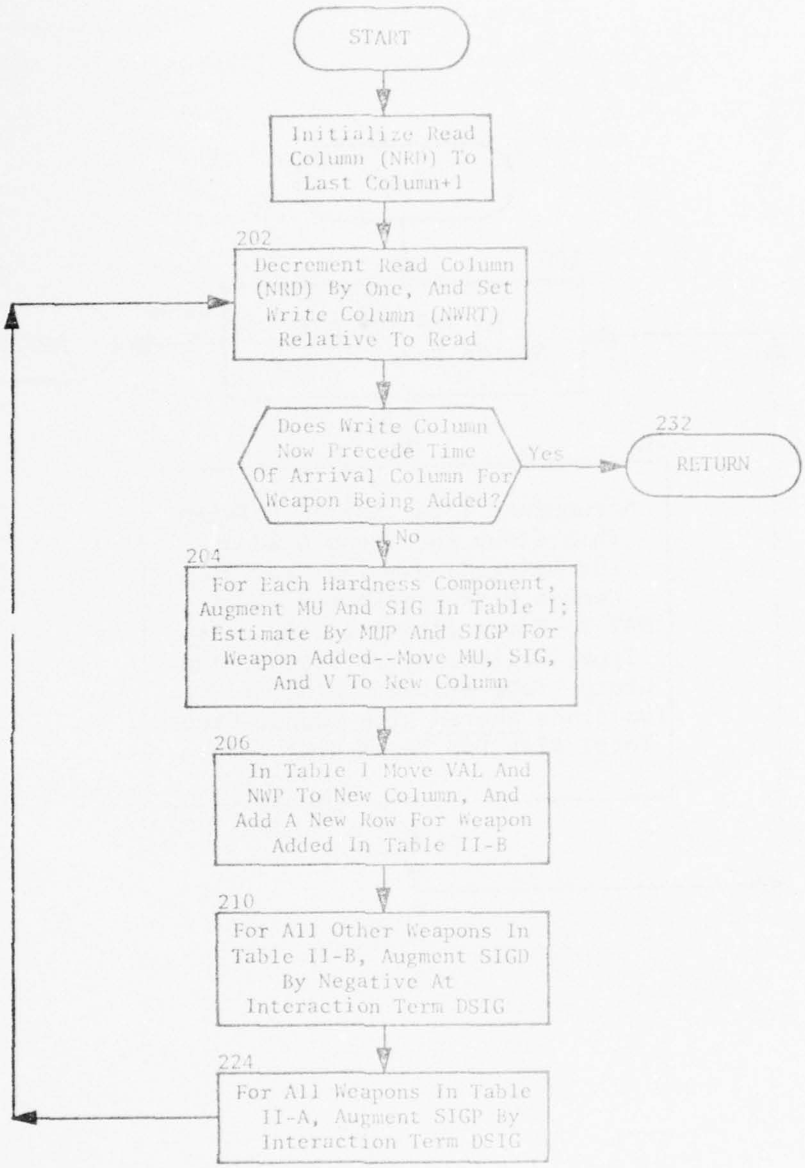


Figure 53. Part VII: Local Subroutine CALSG



This subroutine updates values for MU and SIG in all Time of Arrival columns affected by new weapon G. If weapon G adds a new column, it also moves the data one column to the right by writing in a column with an index one larger than the index of the column being read.

Figure 53. Part VIII: Local Subroutine ADDSIG

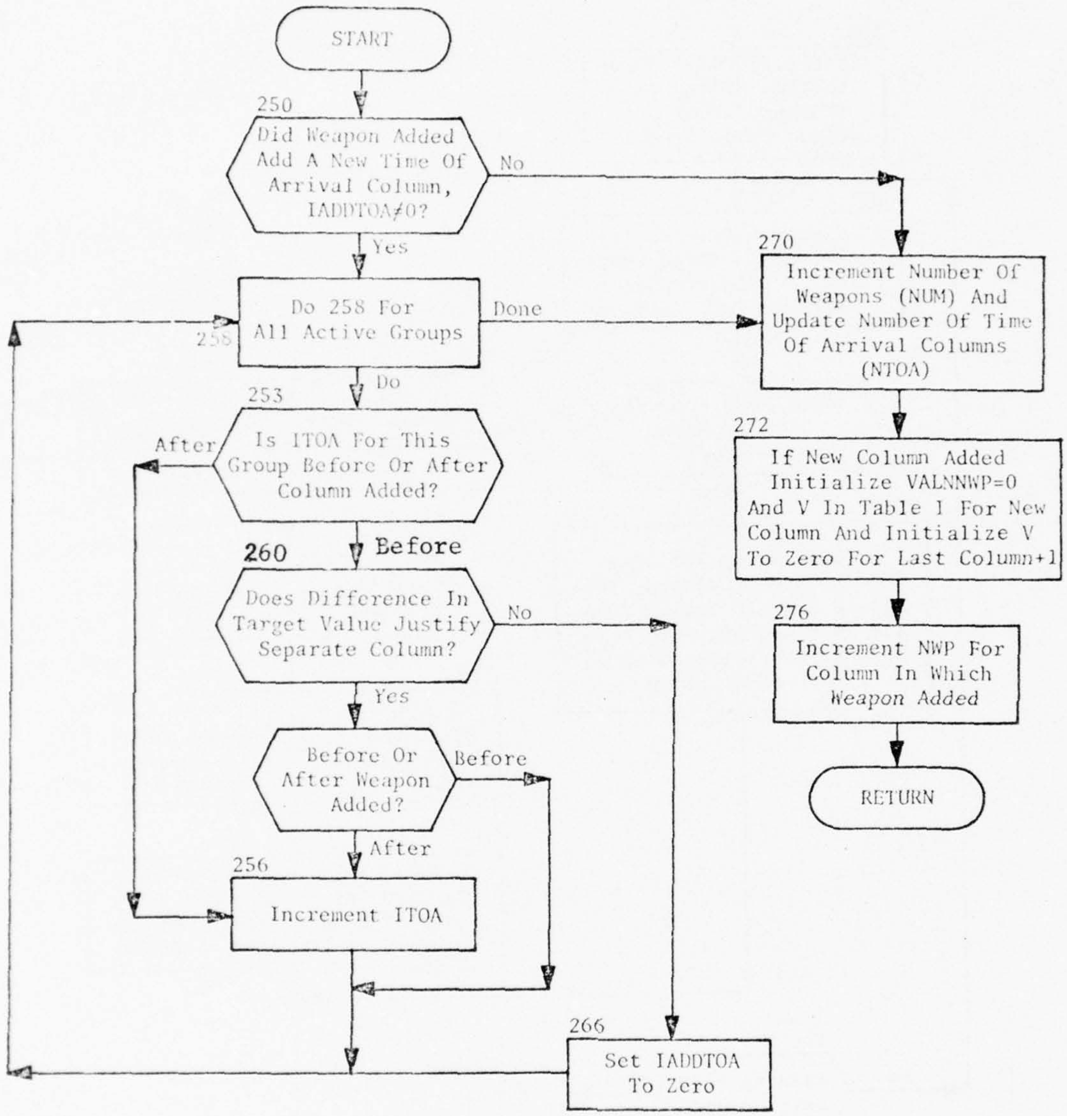


Figure 53. Part IX: Local Subroutine ADDIND

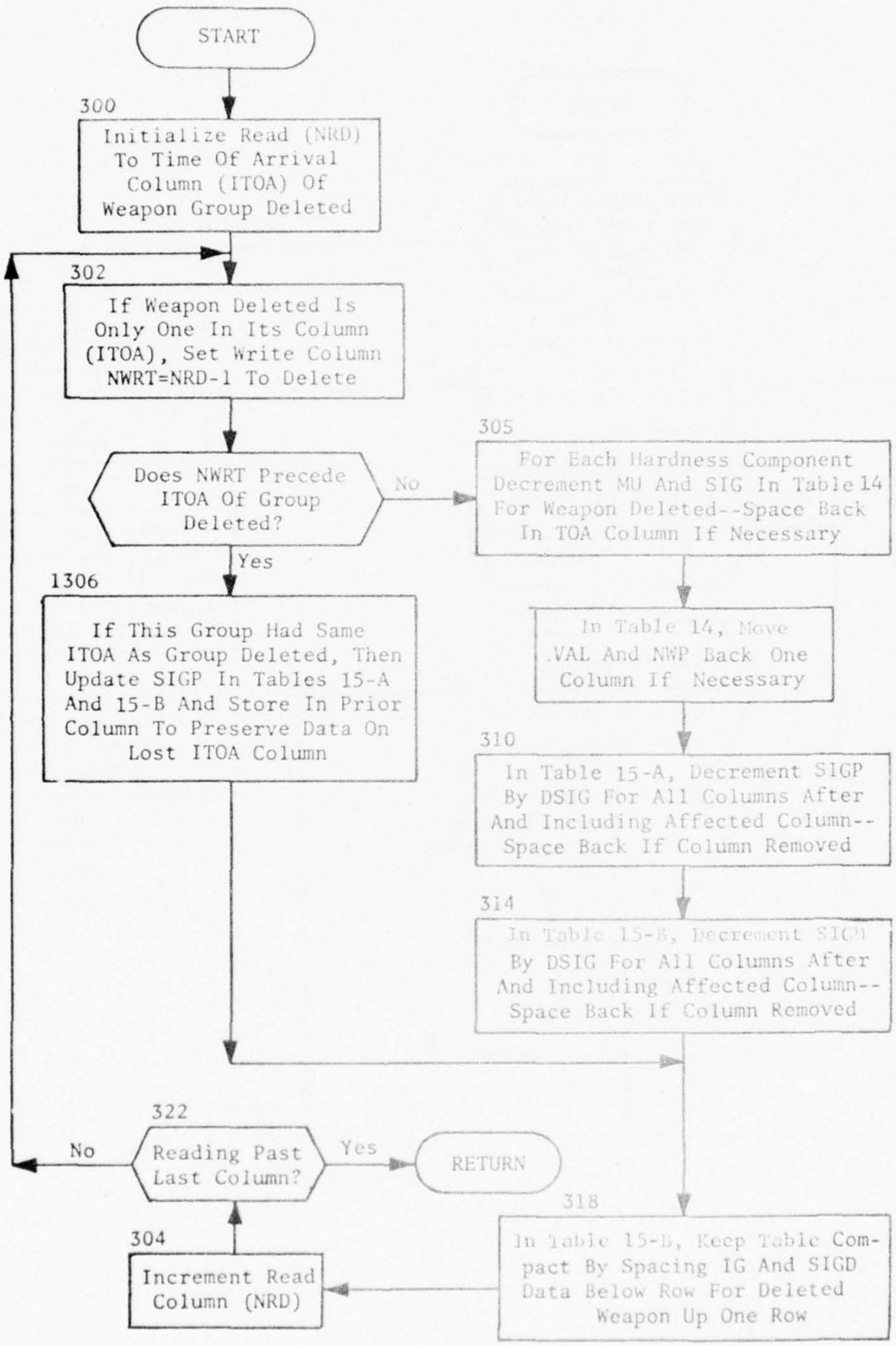


Figure 53. Part X: Local Subroutine SUBSIG

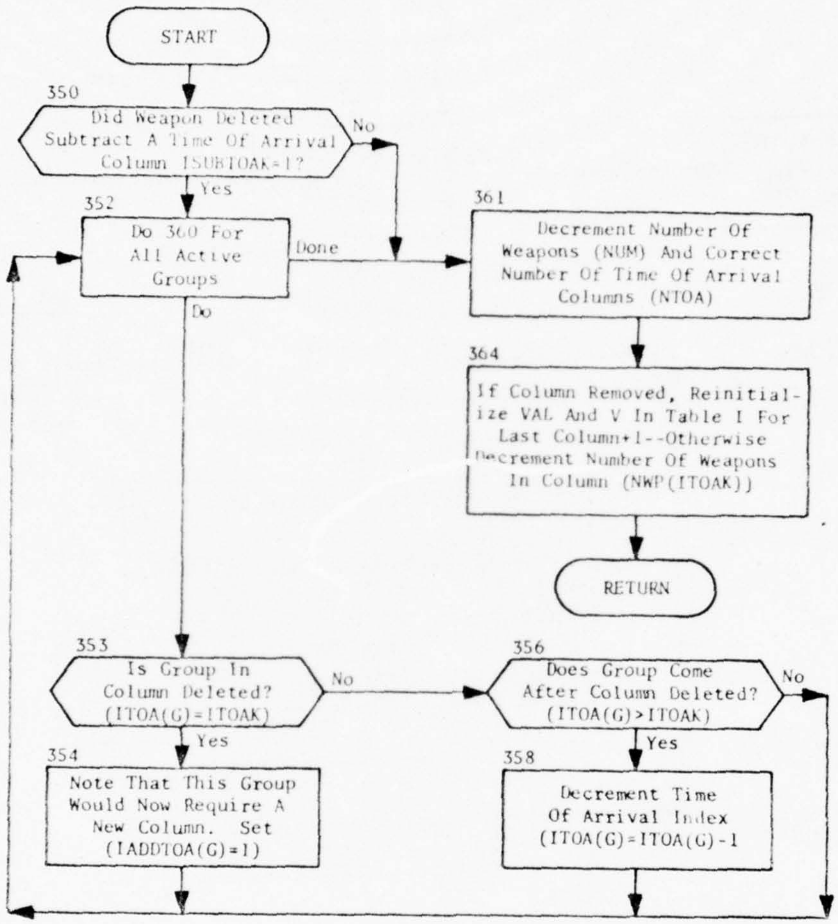
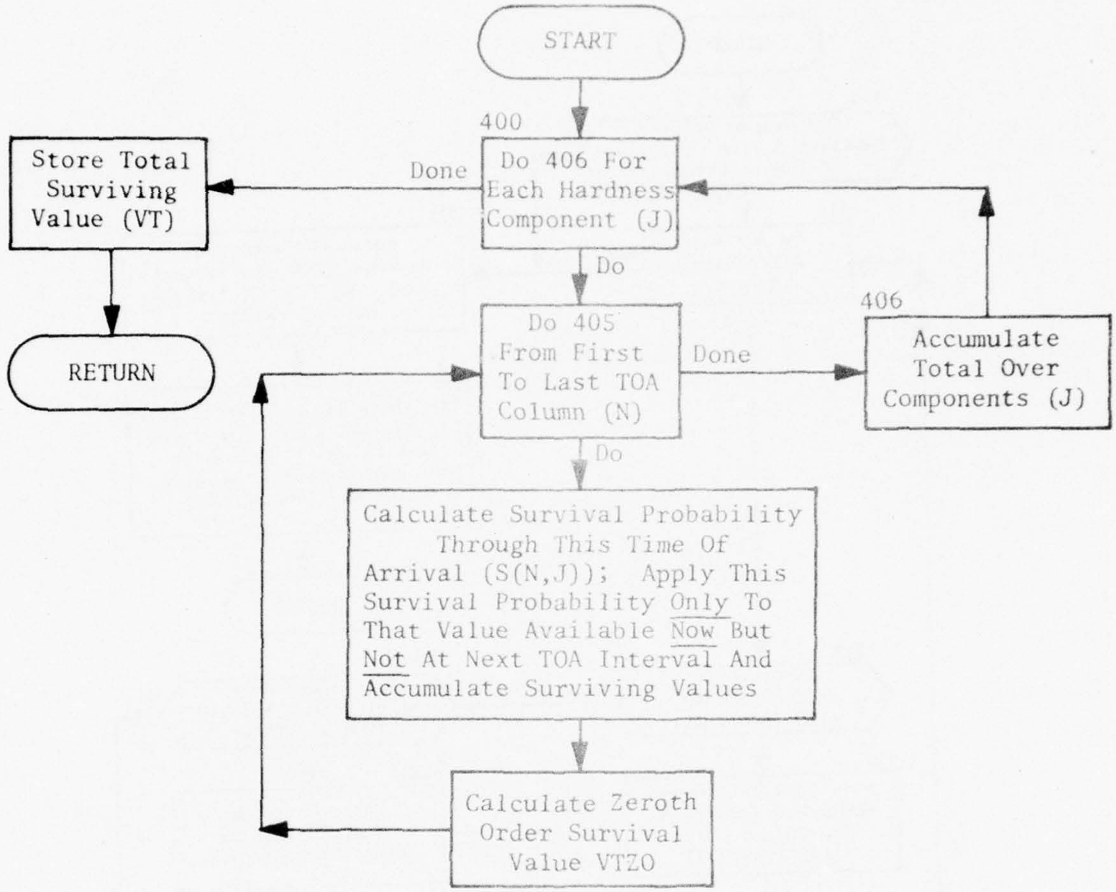


Figure 53. Part XI: Local Subroutine SUBIND



Calculates Actual Surviving Target Value, VT:

$$VT = \sum_{\substack{J= \\ (I,M)}} \sum_{\substack{N= \\ (1,NTOA)}} S(N,J) * [\ V(N,J) - V(N+1,J)\ ]$$

and Zeroth Order Surviving Value, VTZO;

$$VTZO = \sum_{J} S(NTOA,J) * VO(J)$$

Figure 53. Part XII: Local Subroutine CALPAY

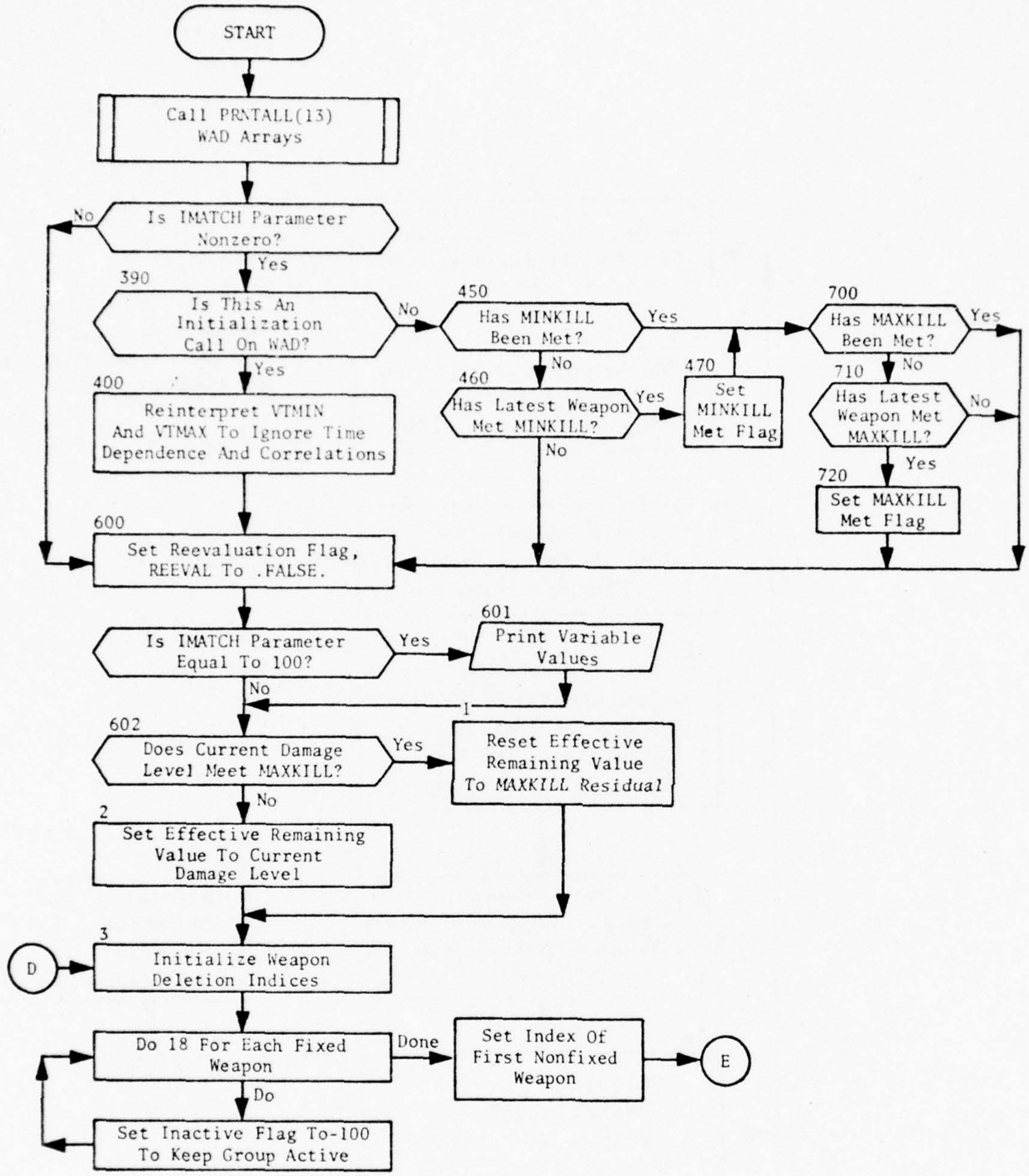


Figure 54. Subroutine WADOUT
(Part 1 of 5)

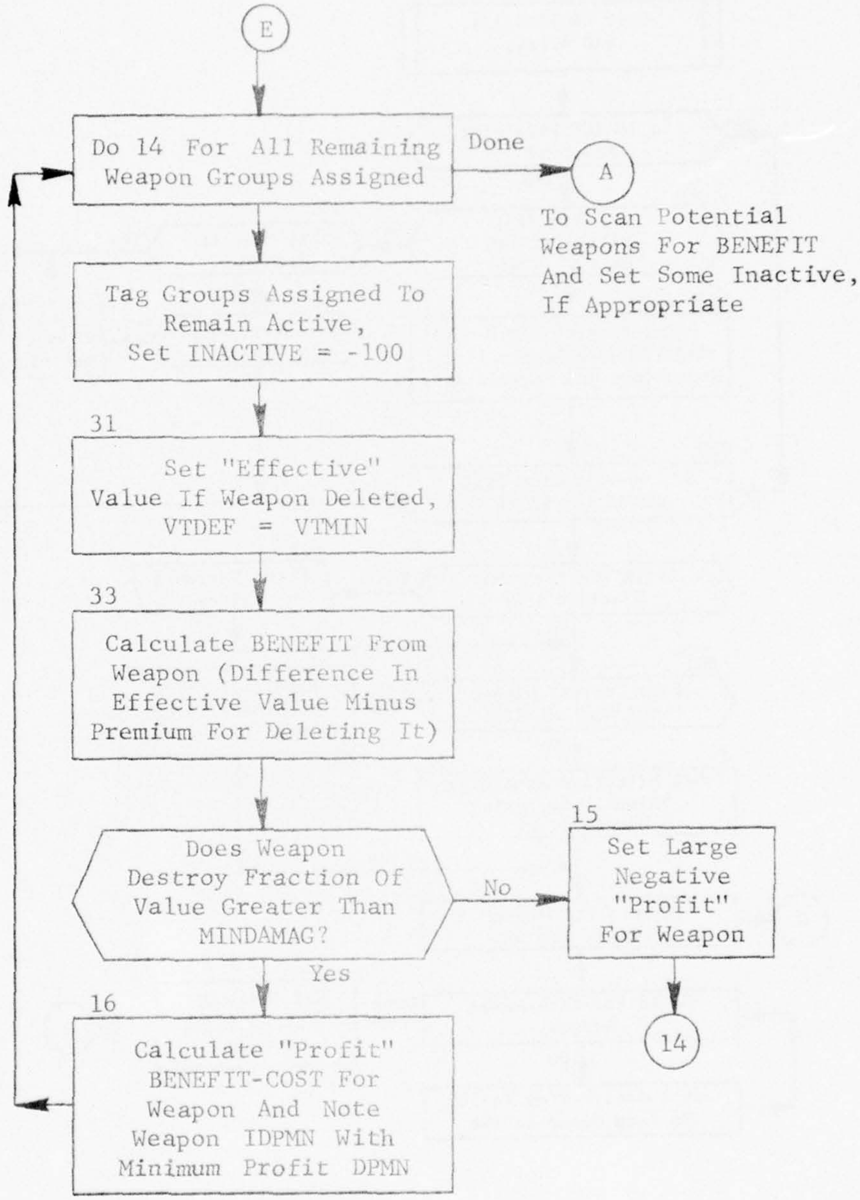


Figure 54. (Part 2 of 5)

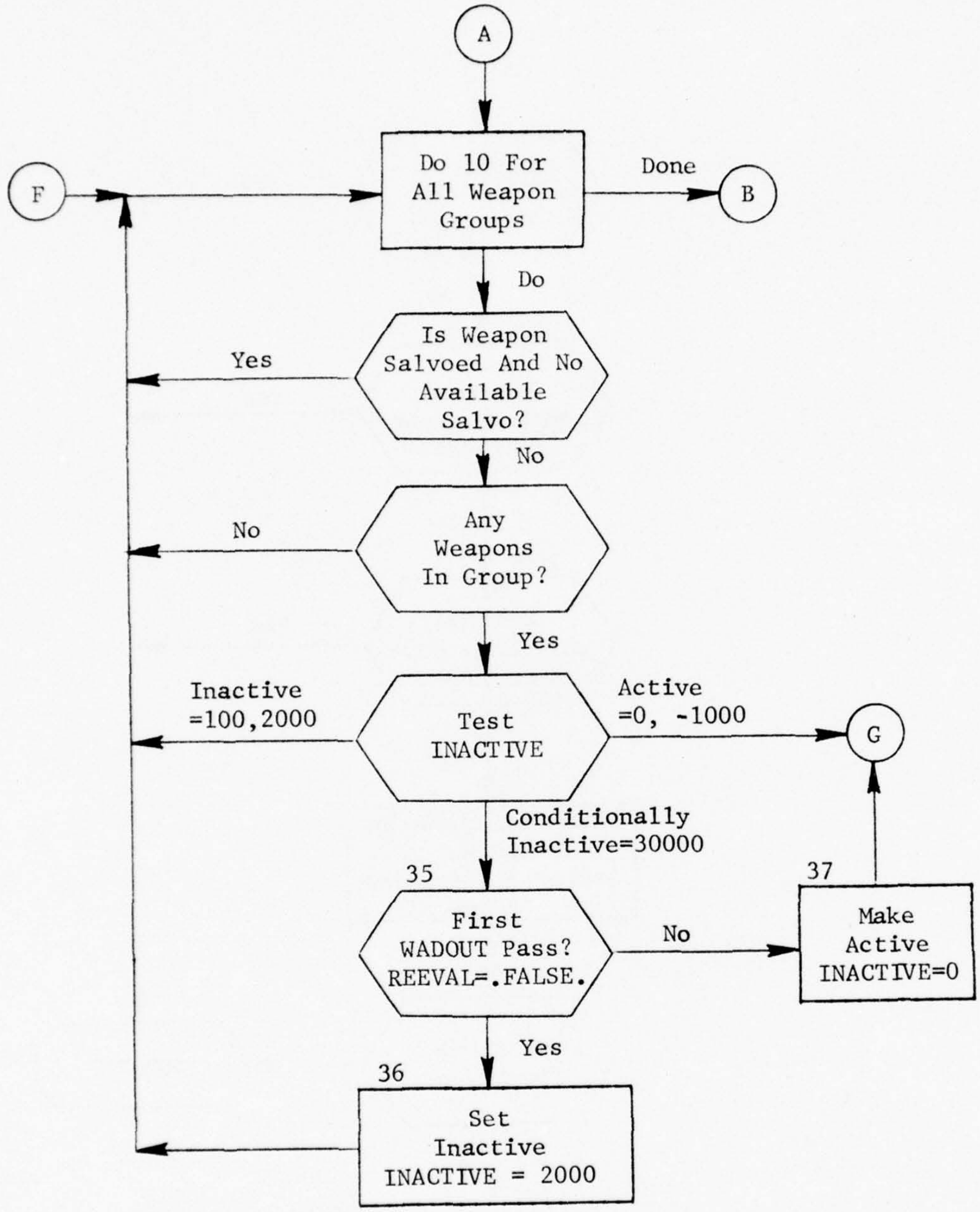


Figure 54. (Part 3 of 5)

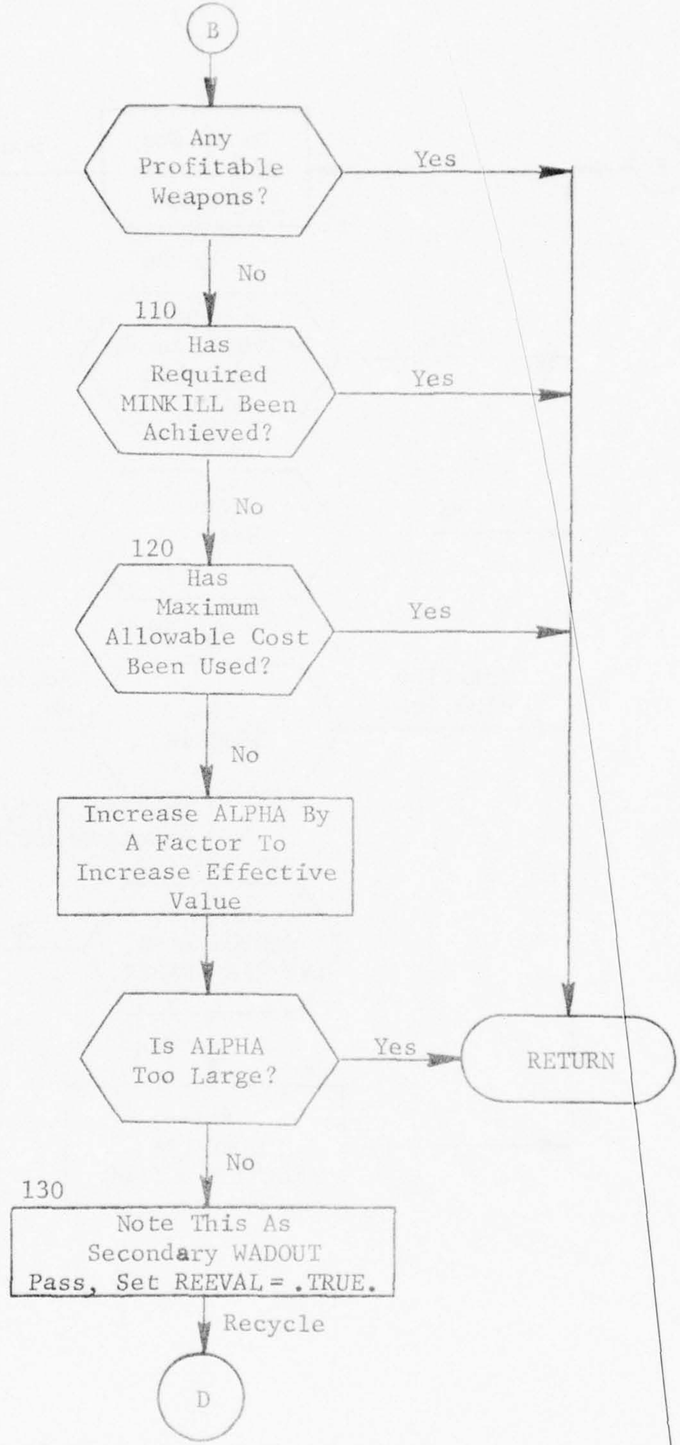


Figure 54. (Part 4 of 5)

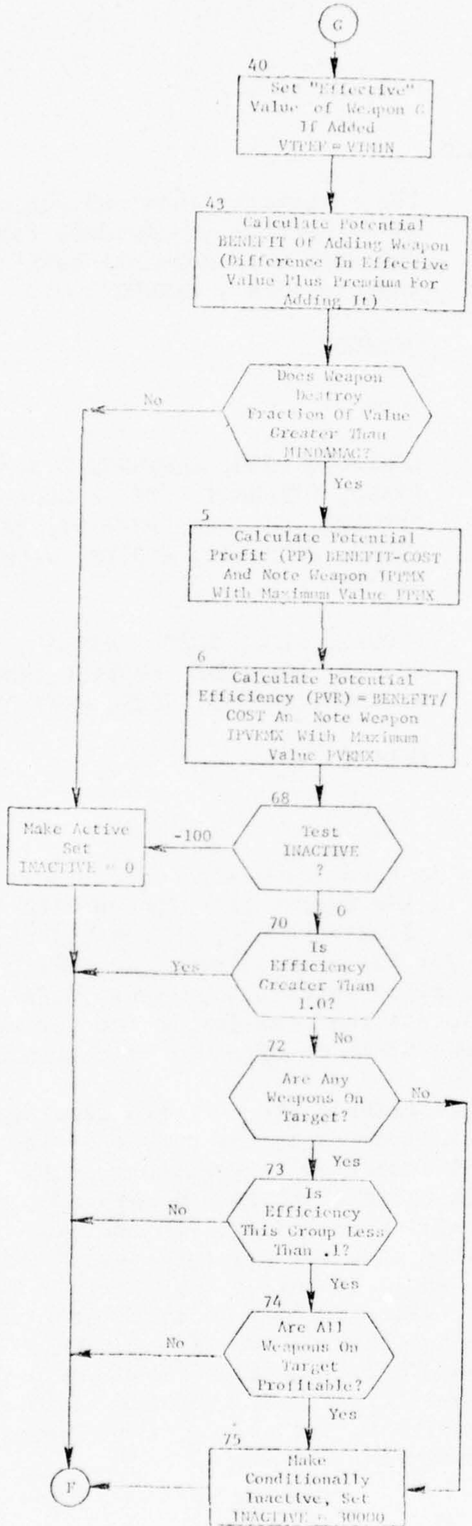


Figure 54. (Part 5 of 5)

3.10 Segment DEFALOC

PURPOSE: The purpose of this subroutine* is to allocate missiles to an individual target which is defended with terminal ballistic missile interceptors (i.e., MISDEF > 0).

ENTRY POINTS: DEFALOC

FORMAL PARAMETERS: None

COMMON BLOCKS: CONTROL, C333, DEFENSE, DYNAMIC, FILES, FILABEL, FIXED, FIXEDASS, ITP, LAMBDA, LOCFIL, MASTER, MYIDENT, MYLABEL, NOPRINT, PAYLOAD, SALVO, TWORD, WADFINAL, WADOTX, WADWPN, WPNGRP, WPNREG WPNTYPE

SUBROUTINES CALLED: ADDSAL, GLOG, IGET, INITSAL, IPUT, LAMGET, NUMGET, PREMIUMS, PRNTALL, RDWORD, RESTORE, RESVAL, SLOG, WRARRAY

CALLED BY: MULCON

Method:

When MULCON has read in data associated with a new target, it examines MISDEF to determine if the target is defended with terminal ballistic missile interceptors. If MISDEF = 0, indicating no defenses, it proceeds to call STALL for the allocation of weapons. If MISDEF > 0, then there are terminal interceptors present; DEFALOC is called after calling STALL to allocate the missiles to the target, and the most profitable allocation (STALL or DEFALOC) is chosen.

The input variables describing the target's local ABM capability allow uncertainties to be introduced in the number of interceptors present. MISDEF is the nominal number of interceptors on the target, each with kill probability PKTX against unhardened warheads, and RADPK is the random area defense kill probability. In addition, four other parameters are defined (the same for all targets) which introduce uncertainties in MISDEF. RX(1) (input as LOWFAC) is a factor which, when multiplied by MISDEF, gives a lower estimate of interceptors which has probability PX(1) (input as PROBLOW) of occurring. Likewise, RX(2) (input as HIGHFAC) and PX(2) (input as PROBHIGH) define the overestimate of interceptor availability. Thus, if there is imperfect knowledge of the terminal ABM capability, the allocator can hedge against these uncertainties when assigning weapons.

---

*This routine is the second segment of the second overlay. It is called via computer system subroutine LLINK.

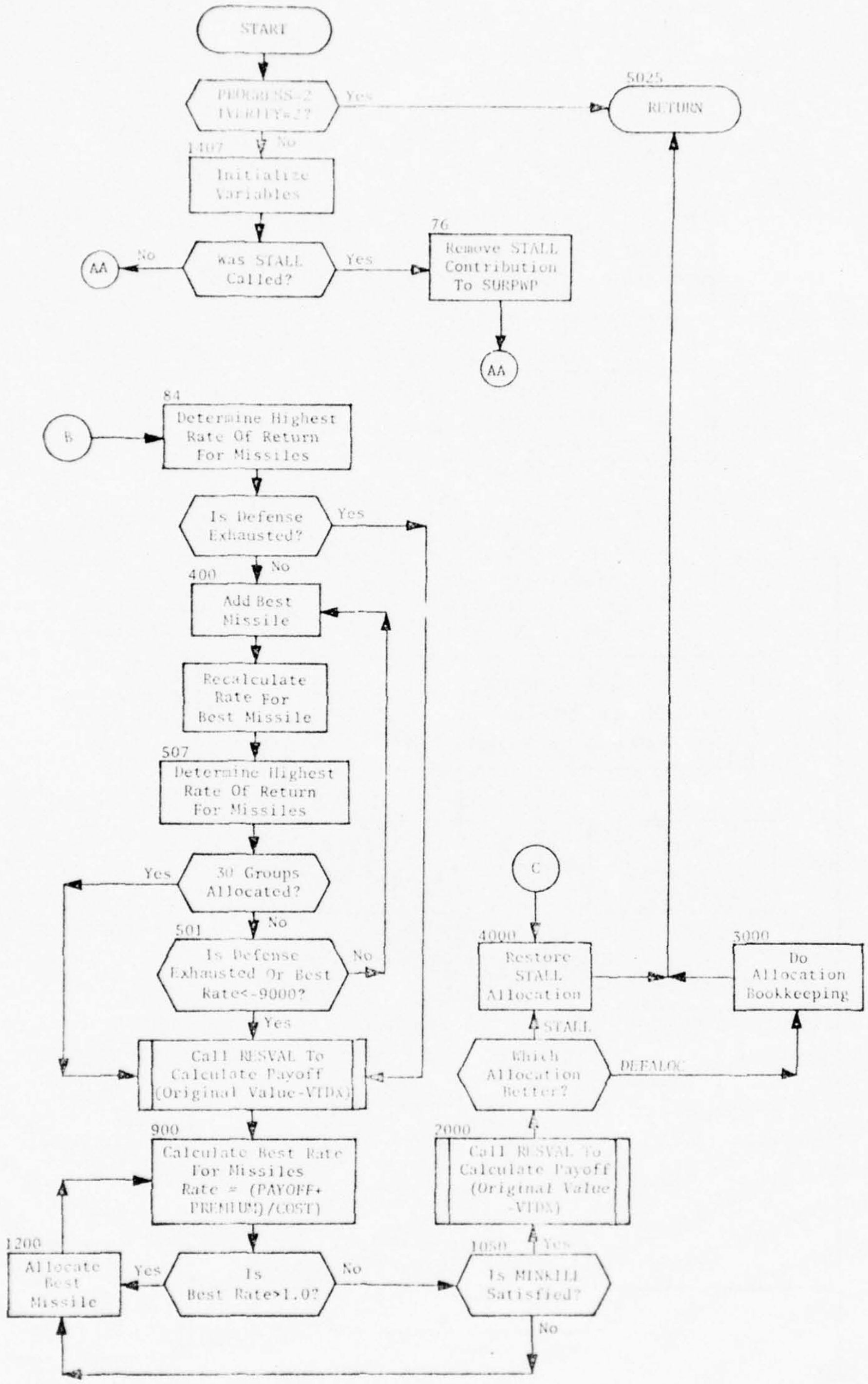


Figure 55. Segment DEFALOC
Part I: Normal Processing
(Part 1 of 3)

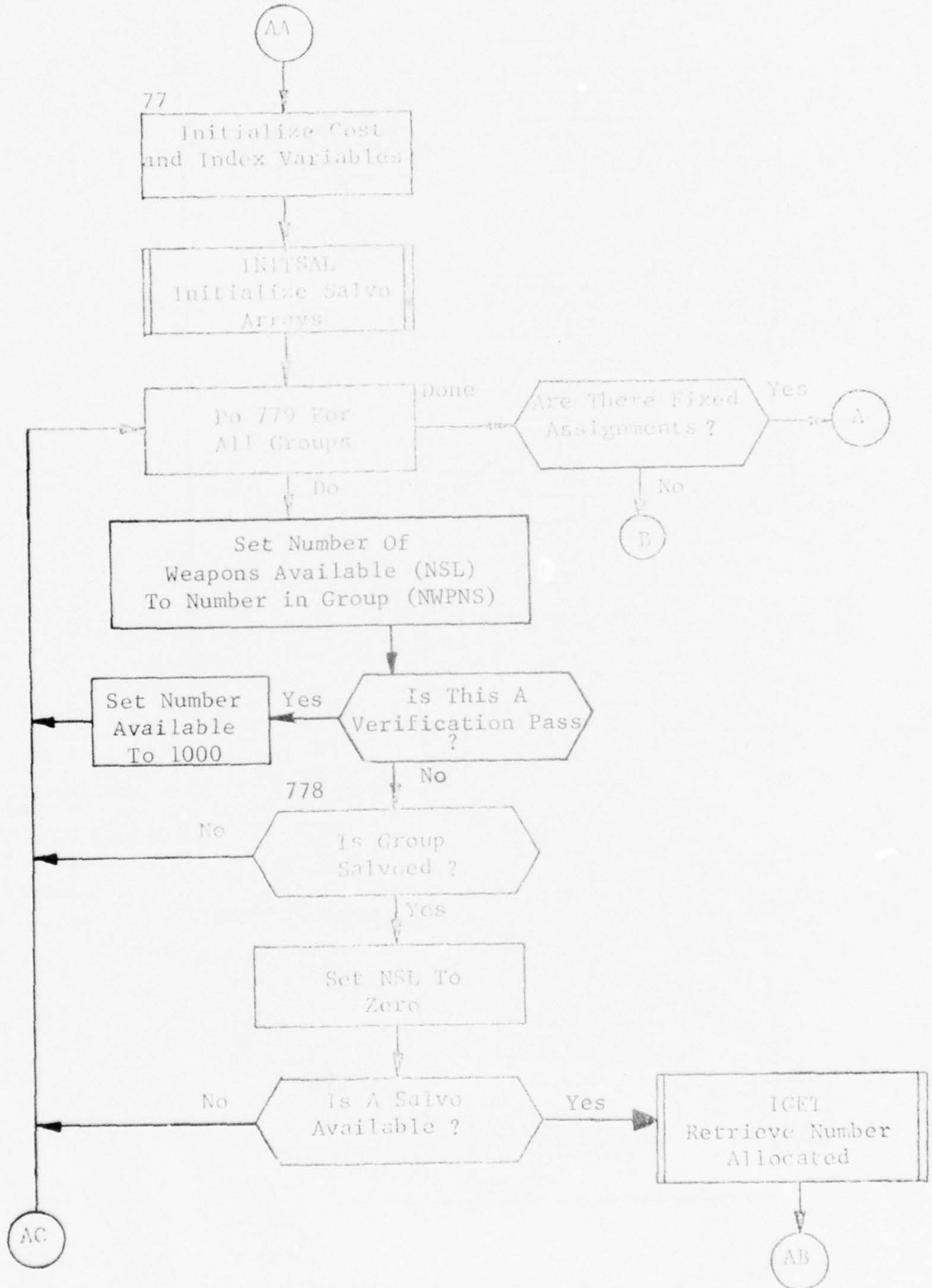


Figure 55. Part I: (Part 2 of 3)

SECTION 4. PROGRAM EVALALOC

4.1 Purpose

The purpose of program EVALALOC is to summarize the planned allocation of weapons to targets and provide an expected-value estimate of the results. Provision is also included to evaluate the allocation for variations in the values assigned selected parameters (planning factors) associated with the weapons and targets. The evaluation can be made for either the whole plan or for only targets in selected countries. EVALALOC may be run at two stages of plan development, either before program ALOCOUT or after program PLANOUT. If run prior to the selection of desired ground zeros (DGZ) for complex targets (accomplished in ALOCOUT), the analysis of aim point offsets is not included. In this case, the results produced by EVALALOC represent an upper limit estimate which assumes that each target element in a complex is directly targeted. When EVALALOC is run after program PLANOUT, the weapon aim points offsets are available and are included in the expected value computations.

4.2 Input Files

When program EVALALOC is run before program ALOCOUT, the input files are the BASFILE prepared by program PREPALOC and the ALOCTAR file prepared by program ALOC. When run after program PLANOUT, the PLANTAPE produced by PLANOUT is also required as input.

4.3 Output File

Program EVALALOC does not produce an output file for use by later processors; its sole output is a set of summaries which present the expected-value results of the planned weapon allocation.

4.4 Concept of Operation

Program EVALALOC processes the targets one at a time. For each target (or target element of a complex target), the assigned weapons are read in and ordered by time of arrival. Surviving target values are calculated, utilizing the same damage functions used in program ALOC (subroutine WAD), except that correlations are ignored. After the survival probability of each target is computed, the target and the assigned weapons are classified for summarization purposes. When all targets have been processed, the expected-value results are summarized and printed.

The initial pass over the target system always produces an evaluation based on the same weapon and target parameters used in program ALOC. Subsequent passes may be made to investigate the sensitivity of the results to changes in these weapon and/or target parameters.

When program EVALALOC is run prior to program ALOCOUT, the weapon allocation data are obtained directly from the ALOCTAR file (ALOCTAR reflects

the allocation by target). When EVALALOC is run in the post PLANOUT mode, the weapon allocation data are obtained from the PLANTAPE but cannot be used directly. The PLANTAPE reflects the allocation by sortie (i.e., each block of data describes a delivery vehicle which transports warheads to one or more targets). Consequently, in this mode of operation, EVALALOC must first process the PLANTAPE and construct from it a file which reflects the allocation by target.

The program consists of the main executive program EVALALOC, a summarizing, data handling, and print subroutine EVAL2, and a computing subroutine EVALPLAN. EVALALOC includes provisions for exploring the sensitivity of the results to the assumed or calculated values of some of the weapon or target planning parameters. It also has the ability to use all targets or any subset of targets, based on country codes, in the evaluation. The program can be recycled and these parameter values can be varied using subroutines WPNMODIF and TGTMODIF.

As indicated above, when EVALALOC is run after PLANOUT, it is necessary to prepare a new file on which the weapon-to-target allocation is target-oriented. Because a large amount of data must be stored to describe the allocation, several items of information must be packed in each word. This is done by the four packing subroutines, PACK, BOMRPAKR, MISLPAKR, and UNPACKER. In the event that the pre-PLANOUT operation is prescribed, the packing routines are never called and the allocations are read from ALOCTAR.

Program EVALALOC (figure 57) reads the user's general control data card and stores the input parameters ITGTMAX, JOPT, and PREFABRT in common block /OPT/; the parameter PKTX in common block /MIS/; and the parameter LAW(1), LAW(2) in common block /LAW/. If ITGTMAX is negative, it stops the run. Otherwise, it initializes the filehandler and reads the /FILES/ and /MASTER/ common blocks from the BASFILE. If the run is post-PLANOUT it also reads the REL, ITYPE, and DBL arrays into common block /CWPNMOD/ so they can be used by subroutine UNPACKER to modify the values of the weapon penetration probability obtained from the PLANTAPE. In this mode, EVALPLAN next calls subroutine PACK to reorder the weapon allocation data from the PLANTAPE. It then calls EVAL2 to evaluate the allocation and summarize the results. In the pre-ALOCOUT mode of processing, subroutine PACK and its associated subroutines are not used, and EVALALOC proceeds directly to the EVAL2 call. After EVAL2 completes its processing, EVALALOC reads the next user general control data card and repeats the process described here until it reads a card where ITGTMAX is negative.

Subroutine PACK is used only in the post-PLANOUT operation of EVALALOC. PACK determines the order of the targets on the ALOCTAR file and reorders the weapon-to-target allocation data on the PLANTAPE in that target order. To do this, it must pack the weapon allocation data using subroutines MISLPAKR or BOMRPAKR (depending on the weapon type). At the time of packing, BOMRPAKR and MISLPAKR use subroutine SEARCH to locate the ALOCTAR target number associated with the target index number, INDEXNO, contained on the PLANTAPE. This number is also packed with the weapon

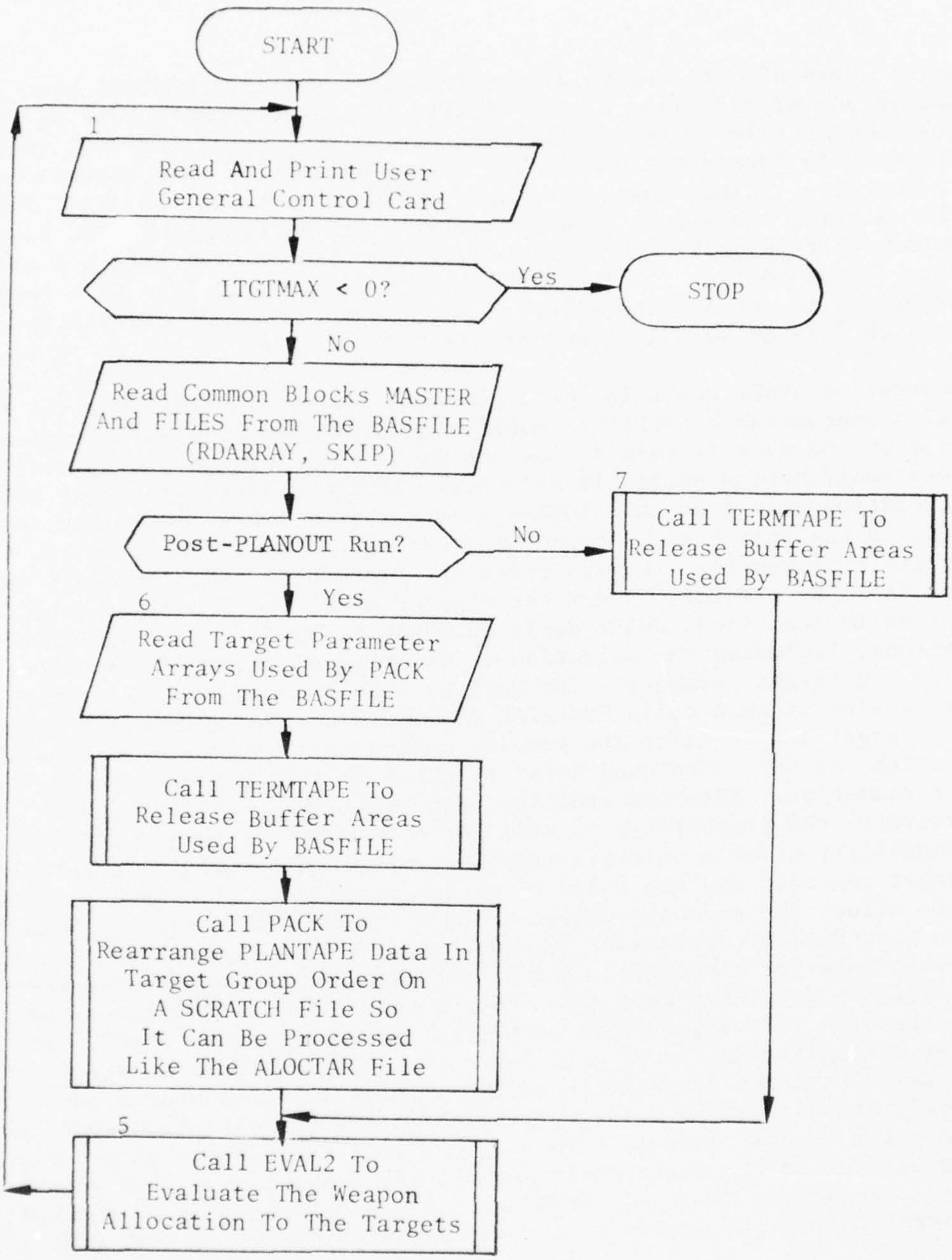


Figure 57. Program EVALALOC

data. When all the weapon allocation data from the PLANTAPE have been packed and written into a scratch file in target number order by PACK, the ALOCTAR file is read again, one target at a time, and PACK uses UNPACKER to locate and unpack the weapon allocation data associated with the targets. Since these data are in target order on the SCRATCH file, the matching process is straightforward. After the allocation data is unpacked, PACK writes the target data from the ALOCTAR file and allocation data (obtained from the PLANTAPE) into a new scratch file in a format identical to the ALOCTAR file. Hence, in post-PLANOUT processing, this scratch file is used in place of the ALOCTAR file.

Subroutine EVAL2 controls the evaluation process in EVALALOC. It first calls subroutine WPNMODIF to modify the BASFILE weapon parameters specified by the user if this is not the first pass through EVAL2, and the first pass modification option is not used. It then reads the target and weapon allocation data from the ALOCTAR file (or from the ALOCTAR-like scratch file in the post-PLANOUT operation) one target at a time. If the target is multiple or complex, it also reads the associated target element data from the BASFILE. If this is not the first pass or the first pass modification option is exercised, EVAL2 calls TGTMODIF to perform user-specified modifications, including the selection or deletion of targets based on country code, on target parameters for each target or complex target component; for all passes it then calls EVALPLAN to determine the amount of damage done to the target and to store the results in the various arrays in common blocks /LITTLE/ or /P/ to be used later in EVAL2 when it produces the printed output summaries. EVALPLAN uses the functions SSKPC, SSSPCALC, and DIST to determine the probability of kill for a target, the target survival probability given a multiple-weapon attack, and the distance between the target centroid and the point of weapon delivery. All three of these factors affect the expected-damage calculation for the target. (Note that in the post-PLANOUT operation, TGTMODIF and EVALPLAN are called once for each multiple-target element.) When all the targets and associated weapon allocation data have been read and evaluated, EVAL2 summarizes and prints the results in tables. This completes one pass through EVAL2. The next pass-through is initiated with another call to WPNMODIF to see if the user wants to test the sensitivity of the results to other weapon parameter modifications. If he does, another pass through EVAL2 is accomplished, including another call to TGTMODIF to select the targets to be used in the evaluation and to modify their parameters. This process continues until EVAL2 encounters the word RETURN as the weapon parameter to be modified. Control is then returned to the main executive program, EVALALOC.

4.5 Common Block Definition

4.5.1 External Common Blocks. The format and content of the external common blocks used in EVALALOC are presented in table 17. The use which the program makes of each of the blocks is briefly set forth below:

a. /FILES/: This common block contains information about the BASFILE, ALOCTAR file, PLANTAPE, and SCRATCH files (which are equivalenced to the STRKFILE which is needed to use the file-handler subroutines).

b. /N/: This common block is filled from the PLANTAPE by subroutine PACK and is used by subroutines BOMRPAKR and MISLPAKR to obtain the data which they pack for each target event.

c. /CTGTMOD/: In post-PLANOUT operation, this block is first filled by PACK when it reads the ALOCTAR file and is then filled by UNPACKER when it unpacks data for the ALOCTAR-like SCRATCH file. Subsequently, in all cases it is filled by EVAL2 when it reads the ALOCTAR file or ALOCTAR-like SCRATCH file. The data are modified by subroutine TGTMODIF and used during EVALPLAN evaluation.

d. /CWPNMOD/: In post-PLANOUT operation, this block is first filled from the BASFILE by EVALALOC so the weapon parameters can be used by PACK to modify the weapon penetration probabilities from the PLANTAPE. In all operations it is filled again from the BASFILE by subroutine EVAL2, modified by WPNMODIF, and used during EVALPLAN evaluation.

e. /PLANREC/: This block is a convenient storage place for data read from the PLANTAPE header portion of each record and is used only by PACK.

f. /MIS/: This block contains data about terminal ballistic missile interceptors and is filled by EVALALOC (PKTX) and EVAL2. It is used by EVAL2 during the calculation of expected target damage when the target has such interceptors.

g. /ASMT/: This block contains the weapon delivery and damage characteristics for the air-to-surface missiles (ASM).

4.5.2 Internal Common Blocks. The format and content of the internal common blocks encountered in EVALALOC are presented in table 18. The use of these blocks by the program is briefly set forth below:

a. /LATLON/: This block is used by subroutines PACK and UNPACKER during post-PLANOUT operation to transfer target and weapon delivery location coordinates back and forth. It is filled in all EVALALOC runs by EVAL2, and EVALPLAN uses it during its calculation of expected target damage.

b. /LITTLE/: This block is filled by subroutine EVALPLAN which uses it to pass to EVAL2 some weapons allocated to one target. EVAL2 updates the arrays used to produce the printed summaries from this data.

c. /OPT/: This block contains the user's options contained on the general data control card. It is used in various ways by nearly all EVALALOC subroutines to carry out the functions specified by the user.

d. /LAW/: This block is filled from the general control data card by program EVALALOC and is used by function SSSPCALC when it determines target survival probability for EVALPLAN.

e. /BINSCH/: This common block is used by PACK, BOMRPAKR, MISLPAKR, and SEARCH and contains parameters related to the binary search done by SEARCH to determine the target number corresponding to a given target index number. It is used in MAKEPRFL and MKTARTAB to hold sortwords, start of data pointers, and scratch array for ordering.

f. /P/: As used by PACK and its associated subroutines, this block contains the packed data from the PLANTAPE. It is used by EVAL2 to store weapon data (CCREL, IREG, IALERT, IPAY, YIELD and PKNAV) from the BASFILE for use by EVALPLAN to hold the arrays in which the summary data for the prints produced by EVAL2 are stored. It is used in MAKEPRFL and MKTARTAB to hold data to be sorted. In MAKEPRFL this data is for the Sample Target List. In MKTARTAB the data is for the Target Designator/Number Directory.

g. /CPF/: This block contains modification flags used by TGTMODIF and EVAL2, and information for the sorts in PRNTFILE and PRTARTAB and for Sample Target List entries in MAKEPRFL.

Table 17. (Part 5 of 5)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
PLANREC (cont.)	MHI(2)	Upper plot markers for sortie
	JHTYPE	Warhead type name
	JPTYPE	Weapon (plan generator) type index
	JFUNC	Weapon function code
	JLAST	End sentinel or not used

Table 18. Program EVALALOC Internal Common Blocks (Part 1 of 7)

BLOCK	VARIABLE OR ARRAY*	DESCRIPTION
ISORTN	ISORTN	Sortie sequence number. This is the first word on a PLANTAPE record
	JSORTN(30)	Temporary storage for sortie sequence number; used in the packing and unpacking routines
LATLON	BLAT	Latitude of a burst event
	BLON	Longitude of a burst event
	TGTLAT	Target latitude
	TGTLON	Target longitude
LITTLE	INDCLAS	Summarization index used for target classes
	SURV	Survival probability of a target
	TGTRAD	Target radius
	TMULT	Original target multiplicity (for multiple targets)
	VALDES	Value of target destroyed
	VALESC	Value of target escaping during attack
	CTMULT	Current target multiplicity (for multiple target)
	PLANYLD	Total yield scheduled to a target
	DELYLD	Total yield delivered to a target
	NOWPNS	Number of weapon categories
	VALFAC	Fractional value of a complex target component referred to total value of complex
OPT	JOPT(3)	Logical array which controls printing and type of operation
	ITGTMAX	Number of targets for which detailed print is given

---

*Parenthetical values indicate array dimensions. All other elements are single-word variables.

Table 18. (Part 2 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
OPT (cont.)	PREFABRT	Probability of refueling abort
	JSKIP	Indicator which contains program control information depending on user's input option
LAW	LAW(2)	Hollerith name of damage function option
BINSCH	NMTGTS	Total number of targets for binary search
	JREAD	Logical indicator equal to .FALSE. on first call to SEARCH, .TRUE. thereafter
	NLOG	Maximum number of iterations in binary search to find target number
	INT	Initial length of binary search interval
	NOW	Initial index of entry in JORDER array to be tested: dummy search
	JORDER(6144)	Array containing words for binary search - each word contains a target index number and its associated target number
BINSCH		As used by MAKEPRFL and MKTARTAB
	SAVE(5)	Not used
	ISORTRA(2048)	The sortwords for each block of information in PL
	ISEQ(2048)	The sequence numbers generated by ORDER and used by REORDER to sort the values in LPTR
	LPTR(2048)	The pointer to the beginning of each data block in PL

Table 18. (Part 3 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
P		As used by PACK, BOMRPAKR, MISLPAKR, UNPACKER
	NOARY	Number of arrays used in the sorting process
	LINK	Index to next words in JLINK, KLINK, LLINK, and SLINK arrays to be packed or unpacked
	ILINK(12000)	Equivalenced to JLINK(3000), KLINK(3000), LLINK(3000), and SLINK(3000)
	JLINK(3000)	Packed array; each word contains target index number, target number, and corridor index
	KLINK(3000)	Packed array; each word contains salvo number (missiles), bomb/ASM indicator (bombers), weapon group number, and burst event latitude
	LLINK(3000)	Packed array; each word contains height of burst indicator, weapon penetration probability, and burst event longitude
	SLINK(3000)	Packed array; each word contains time of burst event and sortie sequence number
P		As used by EVAL2 and EVALPLAN
	CCREL(20)	Command and control reliability (from BASFILE)
	IREG(200)	Weapon region index (from BASFILE)
	IALERT(200)	Weapon alert status; 1 for alert, 2 for nonalert (from BASFILE)
	IPAY(200)	Refuel code for bombers or payload index for missiles (from BASFILE)

Table 18. (Part 4 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
P (cont.)	YIELD(200)	Weapon yield in megatons (from BASFILE)
	CLASTYPE(200)	Summarization index for aggregating types into classes
	NALLTYPE(I,J) (7,200)	Number of weapons of category I scheduled against targets of type J
	SKDWPTYP(I,J) (7,200)	Number of weapons of category I scheduled against targets of type J
	DELWPTYP(I,J) (7,200)	Yield from weapon of category I against targets of type J
	ALLTYPE(I,J) (7,200)	Number of weapons of category I delivered to targets of type J
	NALLCLAS(I,J) (7,50)	Number of weapons of category I scheduled against targets of class J
	SKDWPCL(I,J) (7,50)	Yield from weapon of category I scheduled against targets of class J
	DELWPCL(I,J) (7,50)	Yield from weapon of category I delivered against targets of class J
	ALLCLAS(I,J) (7,50)	Number of weapons of category I delivered against targets of class J
	NAMECLAS(I) (50)	Name of class I
	NOFCLAS(I) (50)	Number of targets in class I
	VOCLAS(I) (50)	Total target value for class I
	VDESCLAS(I) (50)	Time-dependent value of target destroyed in class I
	VESCCLAS(I) (50)	Time-dependent value of target remaining in class I
	VREMCLAS(I) (50)	Time-dependent value of target remaining in class I
	SURVCLAS(I) (50)	Percent of target value surviving in class I
	SKEDCLAS(I) (50)	Megatons scheduled for target class I
	DELCLAS(I) (50)	Megatons delivered to class I targets

Table 18. (Part 5 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
P (cont.)	DESTNOCL(I) (50)	Value of target destroyed in class I
	PCDESTCL(I) (50)	Percent of target value destroyed in class I
	NSUMCLAS(I) (50)	Number of weapons allocated to targets in class I
	SUMCLAS(I) (50)	Number of weapons delivered to targets in class I
	KCLSNGL(I) (50)	Number of targets in class I attacked alone
	KCLSCOMP(I) (50)	Number of targets in class I attacked as part of a complex
	SKEDALL(J) (7)	Total megatonnage scheduled for weapon category J
	DELALL(J) (7)	Total megatonnage delivered from weapon category J
	NAMETYPE(K) (200) . . . KTYPCOMP(K) (200)	The definitions of these arrays exactly parallel those of arrays NAMECLAS through KCLSCOMP, except that they are for target type K (instead of class I)
	NWPNTYP(J) (7)	Total number of weapons scheduled from category J
	XWPNTYP(J) (7)	Total number of weapons delivered from category J
P		As used in MAKEPRFL and MKTARTAB
	PL(12000)	Work area for sorts
HOB	IHOB(90)	Storage for height of burst code
	LXMYHOB	Logical array giving HOB (packed into one word)
KEY	KEYJA	Packing key used for most significant part of target number in (JLINK)
	KEYJB	Packing key used for least significant part of target number (JLINK)

Table 18. (Part 6 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
KEY (cont.)	KEYJC	Packing key used for index number (JLINK)
	KEYJD	Packing key used for corridor index (JLINK)
	KEYKA	Packing key used for salvo number (KLINK)
	KEYKB	Packing key used for group number (KLINK)
	KEYKC	Packing key used for burst event latitude (KLINK)
	KEYLA	Packing key used for height of burst indicator (LLINK)
	KEYLB	Packing key used for weapon penetration probability (LLINK)
	KEYLC	Packing key used for burst event longitude (LLINK)
	KEYSA	Packing key used for burst time (SLINK)
	KEYSB	Packing key used for sortie sequence number (SLINK)
	KEYJOA	Packing key used for most significant part of index number (JORDER)
	KEYJOB	Packing key used for least significant part of index number (JORDER)
	KEYJOC	Packing key used for target number (JORDER)
CPF	ICK	Blank, for simple targets *, for nonrepresentative elements of a complex target **, for the representative target of a complex
	ICOMP	Number of elements in the complex
	MULT	Multiplicity of the target
	NAME	Name of the target
	NCLASS	Class of the target
	JDESIG	Target designator
	NX(52)	ALOCTAR target information
	JSORTN(30)	Sortie numbers on a given target

Table 18. (Part 7 of 7)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
CPF (cont.)	LASTREC(9)	End-of-file record (-1)
	LUNN	Scratch file for sorts
	LUNO	File holding Sample Target List information
	LUNT	Scratch file for sorts
	LTINF	Length of target information
	LWINF	Length of weapon information
	FLAGIST	Flag set first time TGTMODIF is called in a pass
	KEEPCC	Flag showing if a target should be included in the evaluation based on its country code

THIS PAGE INTENTIONALLY LEFT BLANK

AD-A033 653 NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER WASH--ETC F/G 15/6
THE NMCSSC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK) PRO--ETC(U)
NOV 76 R L PAGE, A M PELLICCIOTTO

UNCLASSIFIED NMCSSC-CSM-MM-9-74-V3-CH- NL

2 OF 2
ADA033653

END
DATE
FILMED
2 - 77

4.6 Subroutine BOMRPAKR

PURPOSE:	To pack the weapon allocation data for each bomber target event into four words
ENTRY POINTS:	BOMRPAKR
FORMAL PARAMETERS:	LLL - Index of entry in KPL array which contains the target index number K - Currently not used
COMMON BLOCKS:	HOB, ISORTN, KEY, N, OPT, P
SUBROUTINES CALLED:	IPUT, SEARCH
CALLED BY:	PACK

Method:

BOMRPAKR sets INDX equal to the target number which is KPL(LLL) and uses SEARCH to find the target number ITGT on the ALOCTAR file to which this index number corresponds. Then it determines the index, I, of the next free words in the JLINK, KLINK, LLINK and SLINK arrays where the packed weapon allocation data are stored. The target number ITGT, index number INDEXNO, and the corridor index KORD are packed into JLINK(I). In order to leave bits zero and eight of JLINK(I) set to zero, the target number is split into a most significant part ITA and a least significant part, ITB so that ITGT = ITA*GA + ITB. (Leaving bits zero and eight at value zero is required for subroutine ORDER.) Next, the KLINK(I) word is packed with the salvo number ISAL, the weapon group number LGRP, and the burst event latitude LAT. For bombers, ISAL is set to zero for gravity bombs and one for ASMs. This use of ISAL parallels the use of the ISAL array on the ALOCTAR file. (For missiles, this variable is the missile salvo number.) For packing purposes, the burst event latitude is converted to integer units where ten thousand (10,000) units are one degree of latitude. Word LLINK(I) is then packed with the weapon height of burst indicator IHOB, penetration probability IPEN, and the burst event longitude. IHOB is zero for ground bursts, one for air bursts. The penetration probability is converted to integer units where 4,096 units equal probability 1.0. The burst event longitude is converted to integer units where ten thousand (10,000) units are one degree of longitude. Finally, SLINK(I) is packed with the burst event time ITIME and the sortie sequence number ISORTN. The time is converted to integer units where ten thousand (10,000) units are equal to one hour. Figure 58 illustrates the locations of the packed values.

If the debug print option (JOPT(3)) has been set to TRUE, BOMRPAKR prints all the parameters being packed as well as the packed words, JLINK(I), KLINK(I), LLINK(I), and SLINK(I).

Subroutine BOMRPAKR is illustrated in figure 59.

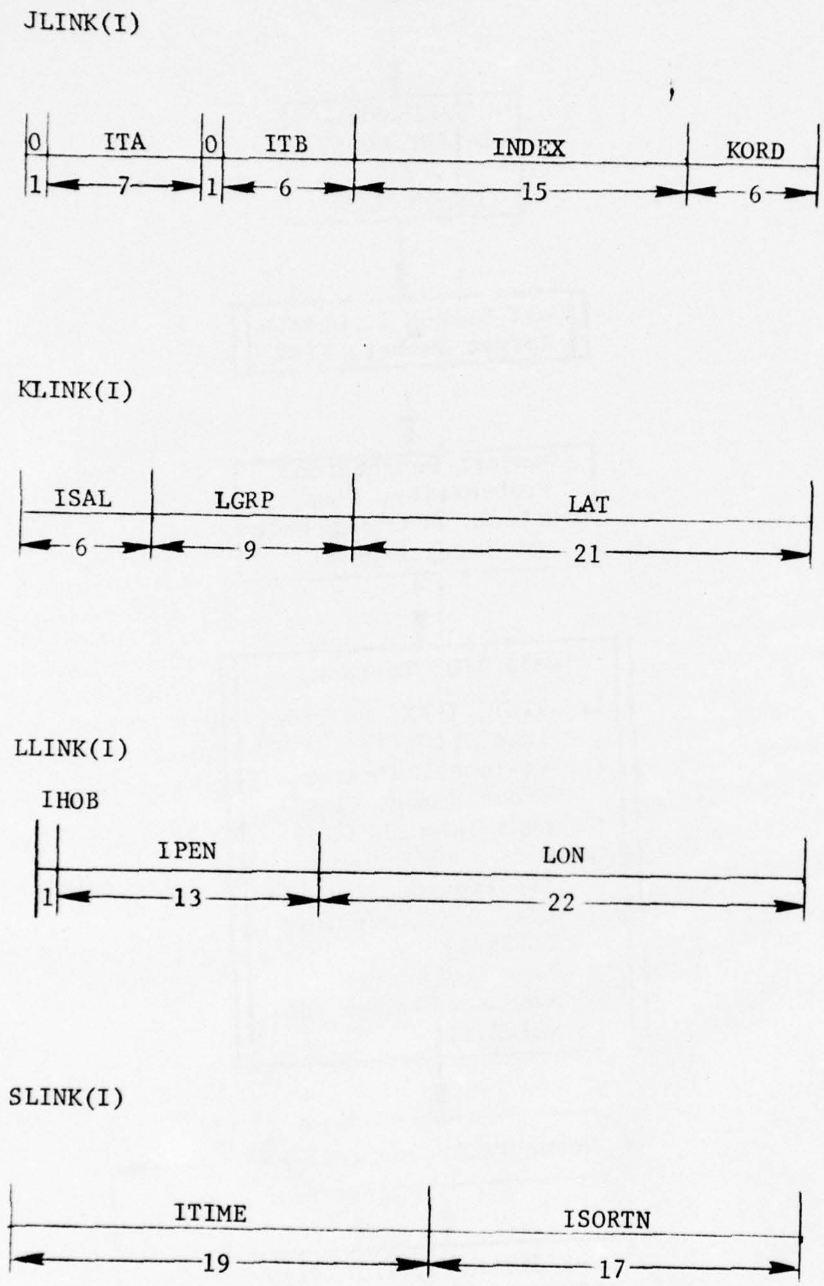


Figure 58. Location of Packed Values
In Subroutine BOMRPAKR

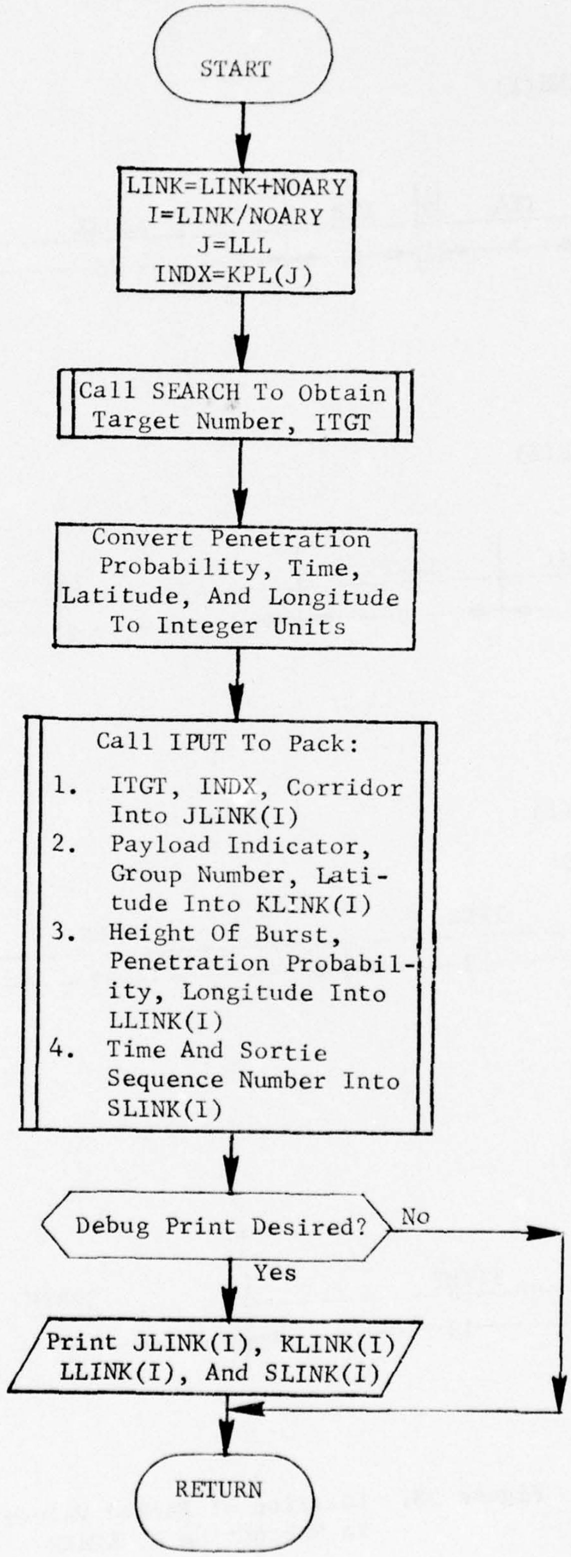


Figure 59. Subroutine BOMRPAKR

THE CONTENTS OF THIS PAGE DELETED

THE CONTENTS OF THIS PAGE DELETED

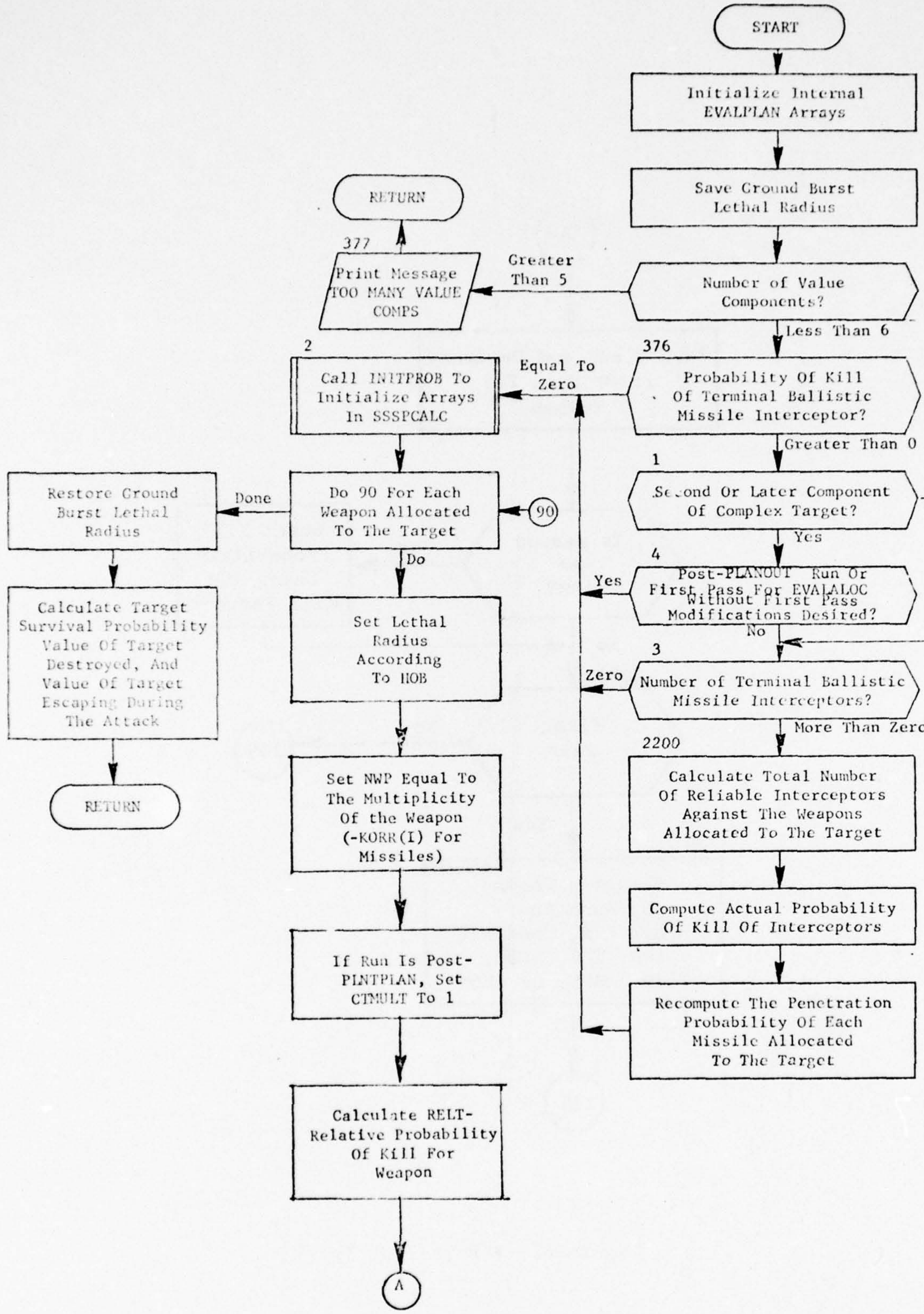


Figure 61. Subroutine EVALPLAN
(Part 1 of 3)

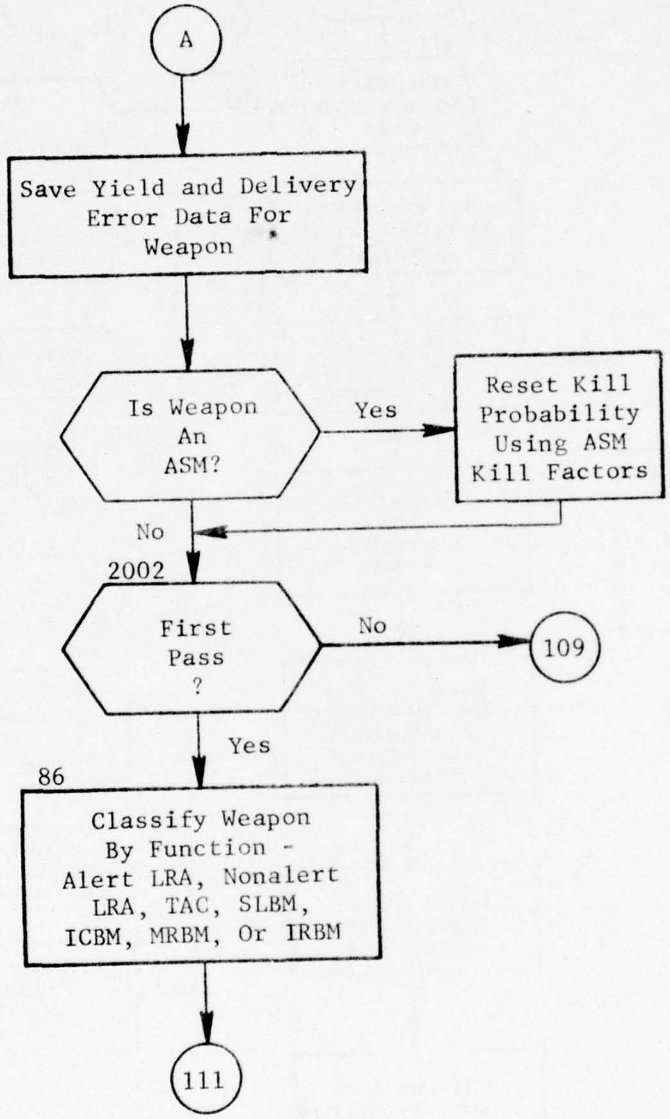


Figure 61. (Part 2 of 3)

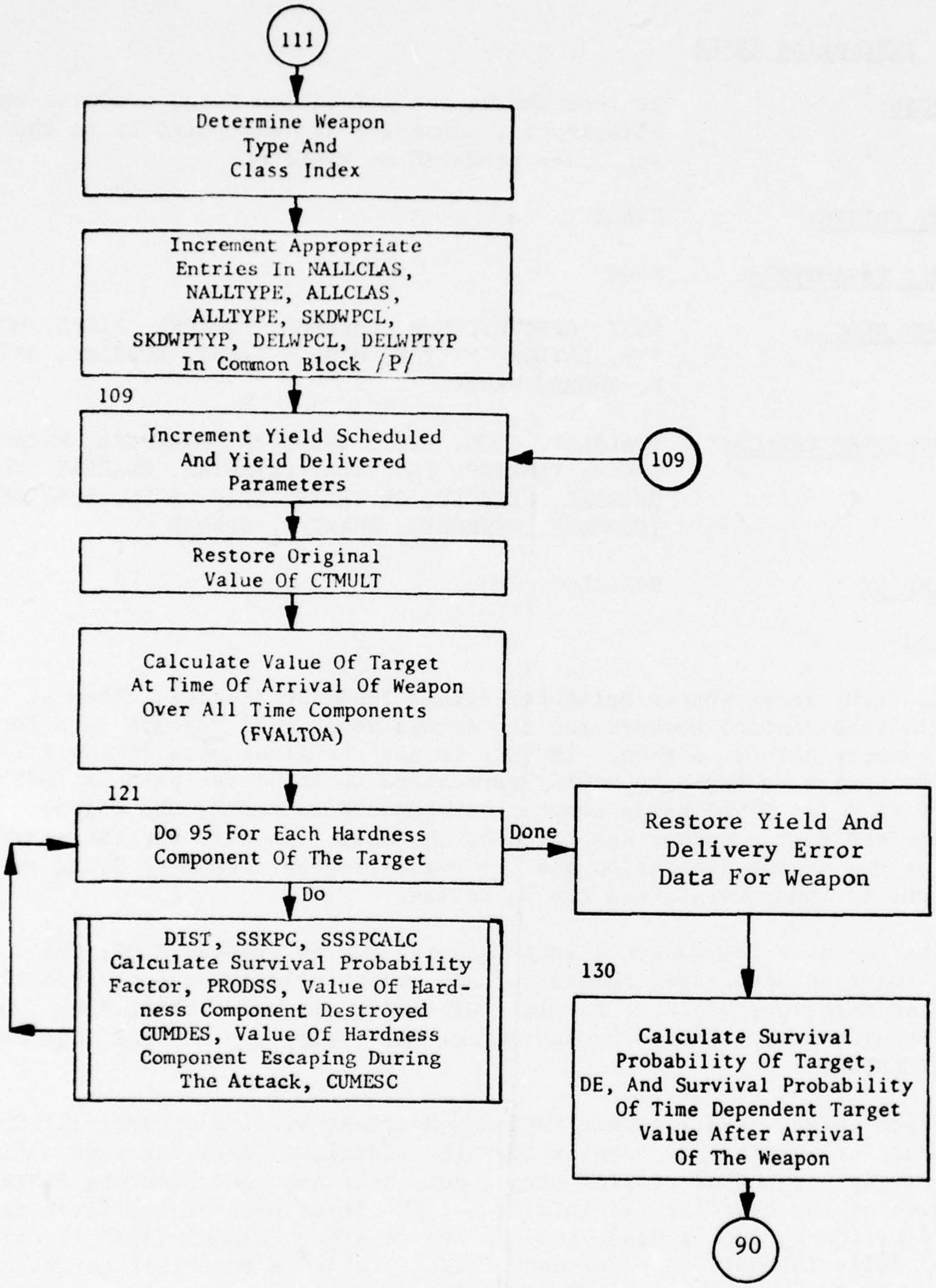


Figure 61. (Part 3 of 3)

4.9 Subroutine EVAL2

PURPOSE: To read the weapon allocation data, evaluate and classify it, summarize it, and print it in the summaries produced by EVALALOC.

ENTRY POINTS: EVAL2

FORMAL PARAMETERS: None

COMMON BLOCKS: ASMT, CPF, CTGTMOD, CWPNMOD, FILABEL, FILES, HOB, ITP, LATLON, LITTLE, MIS, MYIDENT, NOPRINT, OPT, P, TWORD, WAROUT

SUBROUTINES CALLED: EVALPLAN, GLOG, ITLE, LREORDER, MAKEPRFL, MKTARTAB, ORDER, PAGESKP, PRNTFILE, PRTARTAB, RDARRAY, RDWORD, REORDER, SETREAD, SETWRITE, SKIP, SLOG, TERMTAPE, TGTMODIF, WPNMODIF, WRARRAY, WRWORD

CALLED BY: EVALALOC

Method:

EVAL2 first reads weapon parameter arrays from the BASFILE. Then it initializes control numbers and the arrays which will contain data for the summary prints to zero. If this is not the first pass (NPASS $\neq$ 1) or the option to begin to modify parameters on the first pass is indicated (JOPT(2) = 1), EVAL2 calls subroutine WPNMODIF to modify the weapon parameters in the manner specified by the user. In this way the sensitivity of the weapon allocation and the evaluation performed by EVAL2 to changes in these parameters can be tested.

If the run uses the weapon allocation prepared by program ALOC, EVAL2 then reads and processes the target and weapon allocation data from the ALOCTAR file, one block at a time. Otherwise, it reads these data from the SCRATCH file prepared by subroutine PACK using the ALOCTAR file and the PLANTAPE.

For each target, EVAL2 orders the weapon arrays by time of arrival; in the case of multiple or complex targets, additional data for each multiple target element or complex target component are read from the POSTDATA section of the BASFILE. If this is not the first pass or the first pass modification option is used, and the run is after PLANOUT (JOPT(1) =1), EVAL2 calls TGTMODIF once for each target, target element, or target component to perform user-specified modifications on the target parameters; again, this is done to test sensitivity of the evaluation to these parameters. TGTMODIF also determines if the target is to be used in the evaluation based on its country code. It also calls EVALPLAN for each target, target component to classify the weapons allocated, the expected damage done by the weapons, and the change in value of the target.

This process is repeated for each target on the ALOCTAR or SCRATCH file. Two differences exist if EVALALOC is run before ALOCOUT instead of after PLANOUT: First, EVAL2 calls TGTMODIF and EVALPLAN only once for each multiple target in this case. Second, if this is the first pass, it writes the target index number and the ALOCTAR weapon arrays on a scratch file which is used in subsequent passes by subroutine TGTMODIF to perform its target parameter modifications. During this phase of EVAL2 processing, the target and weapon data are saved for a SAMPLE TARGET LIST.

When all target and weapon data from the ALOCTAR or SCRATCH (post-PLANOUT run) file have been processed, PRNTFILE is called to sort the data for the SAMPLE TARGET LIST into Region, Country Code, Designator order and prints out up to the user specified number of targets in a SAMPLE TARGET LIST. EVAL2 then prepares summary tables which describe the results of the allocation. These cumulative results of the weapon allocation are stored and printed by target class and type. The TARGET DESTRUCTION SUMMARY is produced during every pass through EVAL2. (EVAL2 does a new evaluation of the weapon evaluation for each set of user weapon and target parameter modification cards; each evaluation is a new pass through EVAL2.) The other summaries are produced by EVAL2 only during the first pass. They are the SCHEDULE OF WEAPONS ALLOCATED, the SCHEDULE OF WEAPONS DELIVERED, SCHEDULED MEGATONNAGE, DELIVERED MEGATONNAGE, and the TARGET DESIGNATOR/NUMBER DIRECTORY. When all passes through EVAL2 are completed, control returns to EVALALOC.

Subroutine EVAL2 is illustrated in figure 62.

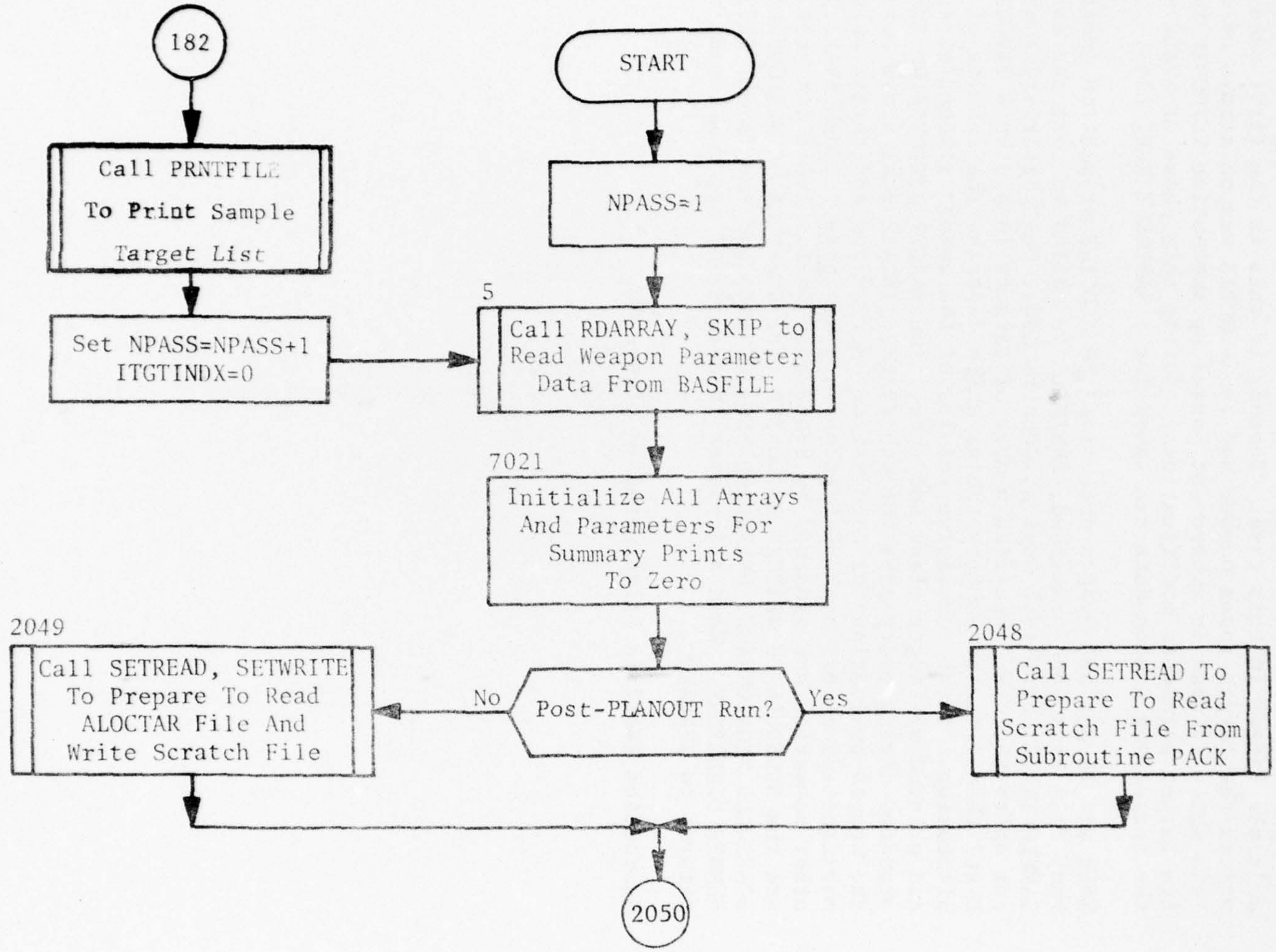


Figure 62. Subroutine EVAL2
(Part 1 of 7)

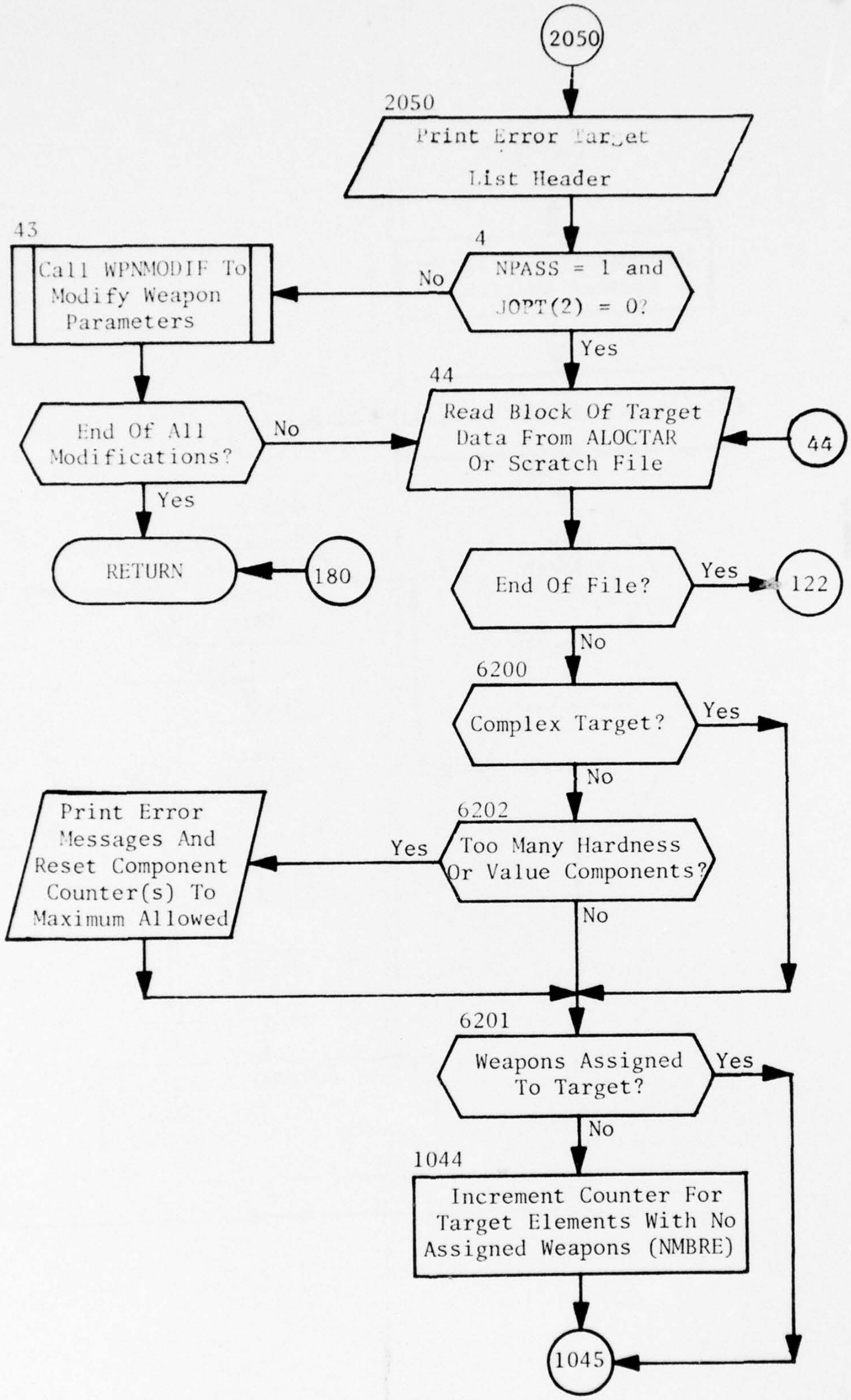


Figure 62. (Part 2 of 7)

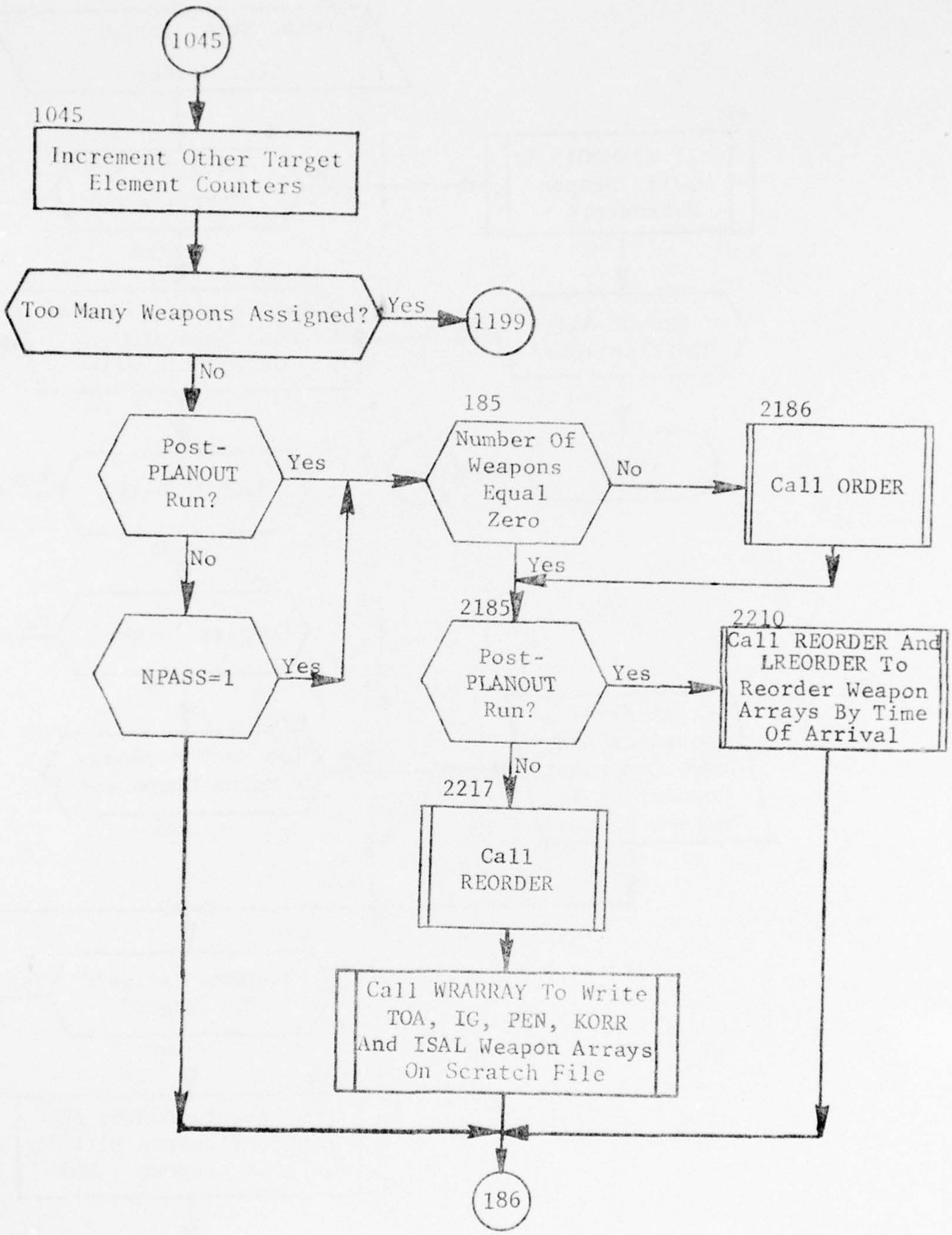


Figure 62. (Part 3 of 7)

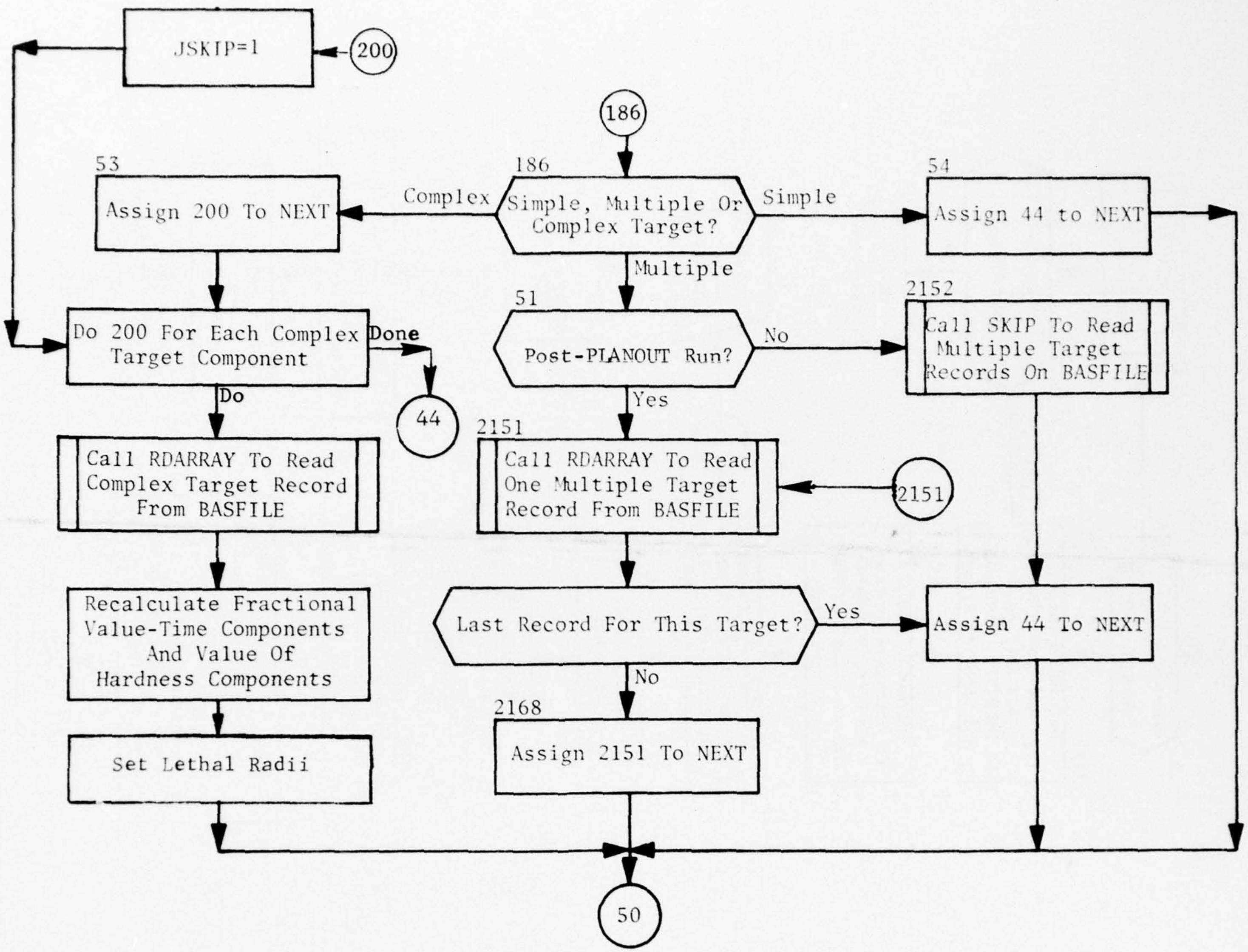
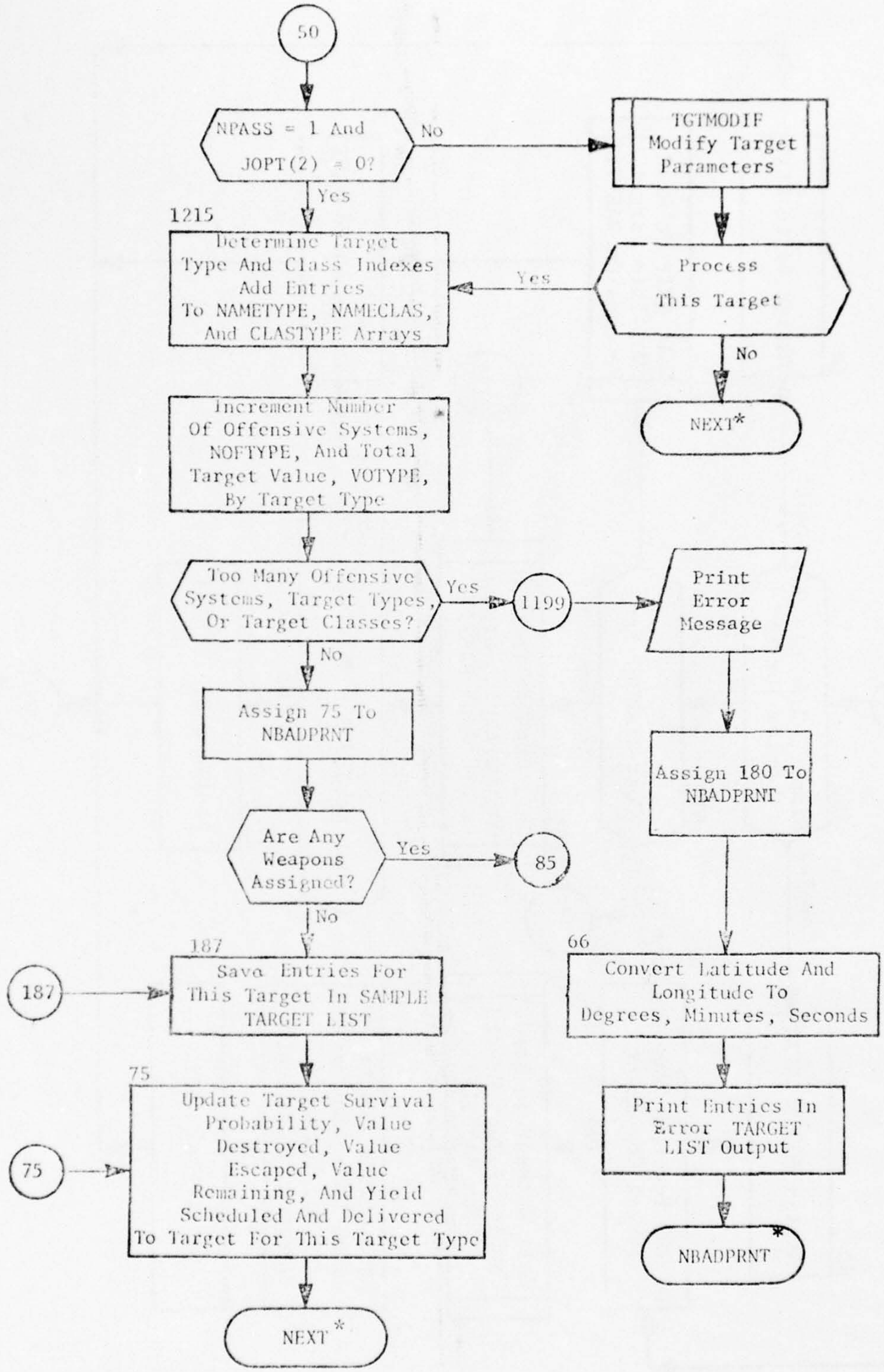


Figure 62. (Part 4 of 7)



*Standard FORTRAN assigned GO TO.

Figure 62. (Part 5 of 7)

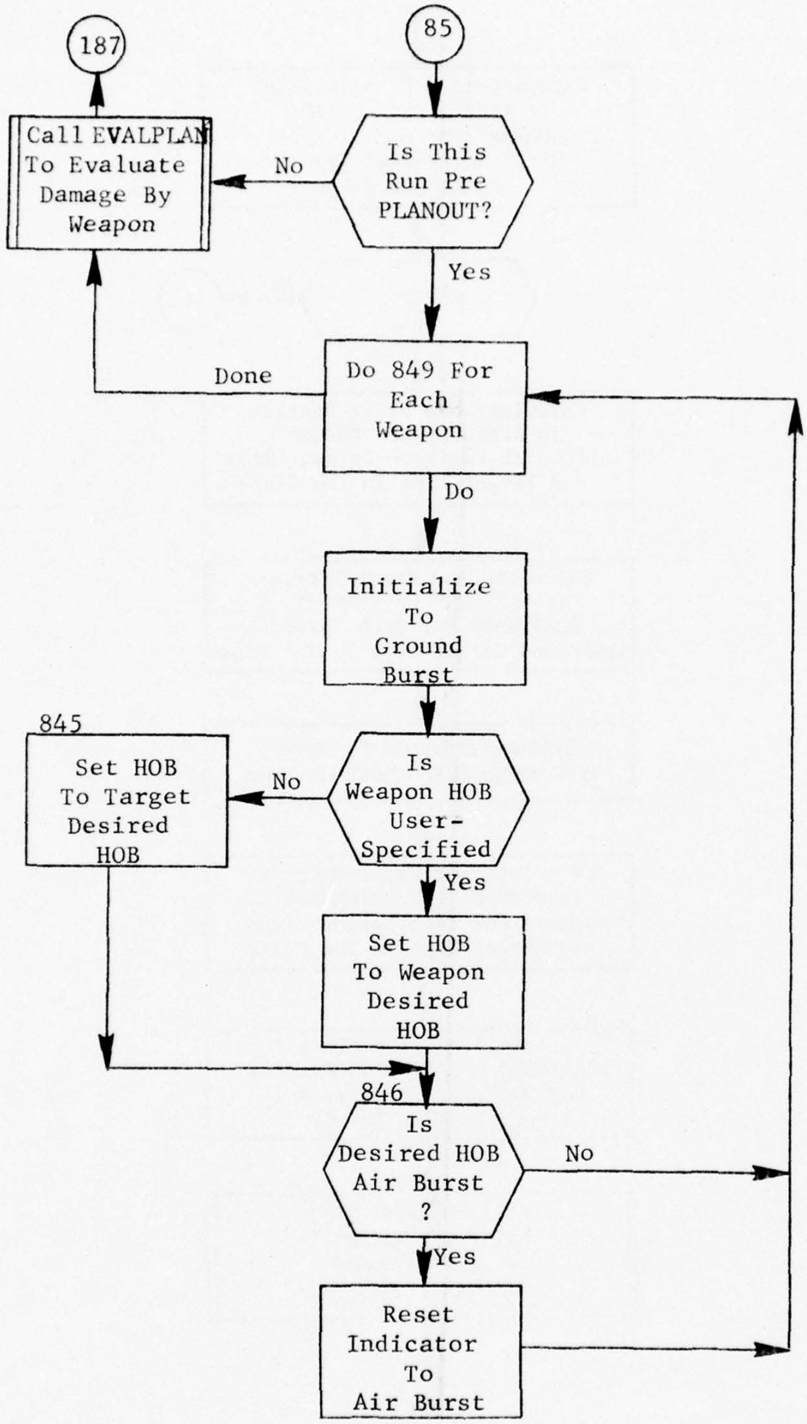


Figure 62. (Part 6 of 7)

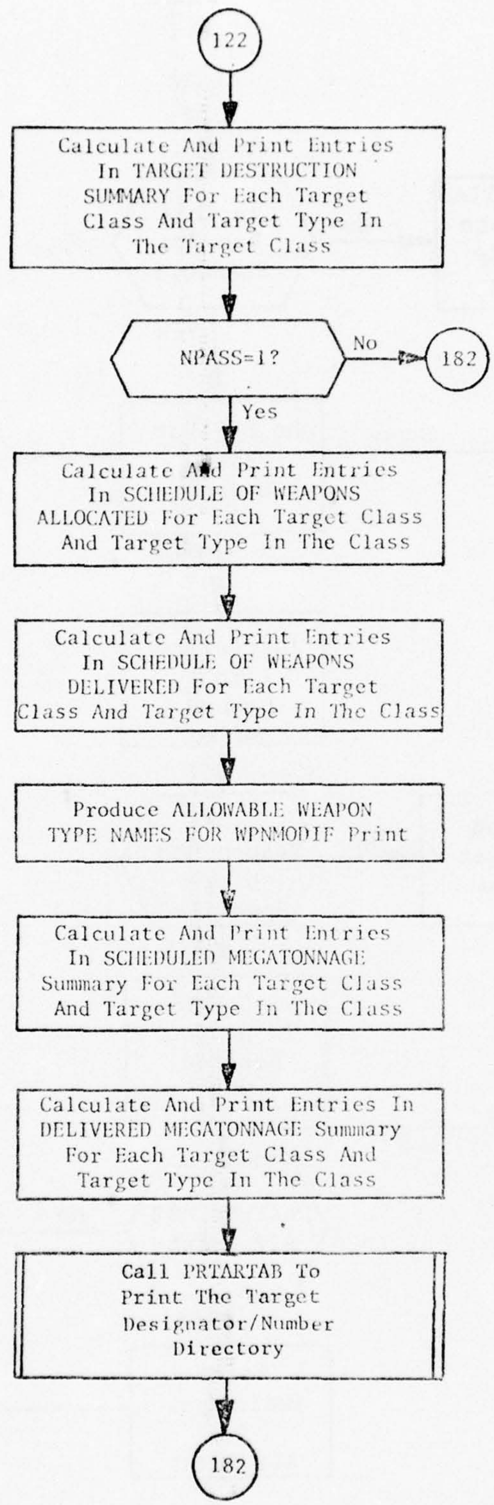


Figure 62. (Part 7 of 7)

4.9.A Subroutine MAKEPRFL

PURPOSE: To sort and print the SAMPLE TARGET LIST.

ENTRY POINTS: MAKEPRFL, PRNTFILE

FORMAL PARAMETERS: None

COMMON BLOCKS: ASMT, BINSCH, CPF, CTGTMOD, HOB, ITP, LATLON, MYIDENT, OPT, P, TWORD, WAROUT

SUBROUTINES CALLED: GLOG, IGET, IPUT, KEYMAKE, ORDER, RDARRAY, REORDER, SETREAD, SETWRITE, TERMTAPE, WRARRAY

CALLED BY: EVAL2

Method:

Subroutine EVAL2 processes each target in the order of occurence on the ALOCTAR file which is a random sequence defined in program PLANSET. Since this order is random, it was desirable to write MAKEPRFL to sort and print the detailed target information in a fashion more meaningful to the user. Currently, this routine will sort and print target information by target's Region, Country Code, and Target Designator. Entry MAKEPRFL is called as each target is processed by EVAL2 to simply store the packed sortword and packed target and weapon information onto a scratch disk file defined by parameter LUNO. At the end of any given pass, entry PRNTFILE is called to read file LUNO, sort according to the sortword, and print the sorted information.

Each time MAKEPRFL is called, only the information to be printed is packed and written onto scratch file LUNO. This packed information consists of one word defining the sortword followed by eight words of target information and further followed by five words for each weapon allocated to the target; see figures 62.1 and 62.2. For a complex target, the sortword for the representative target is used for the entire target complex. Elements of a complex, then, will follow the representative element but only the representative will be in sort order.

At the end of each pass PRNTFILE sorts the file created in MAKEPRFL according to the target's Region, Country Code, and Designator that are packed into the first word of each target information. In addition to scratch file LUNO, two other scratch files labeled LUNN and LUNT are used. LUNT holds the information from LUNO that is in sequence and LUNN receives the output from LUNO. Four working arrays are required: PL holds the packed target and weapon information, ISORTRA contains the sortword of each record, LPTR contains pointers linking the sortwords with starting locations of information in array PL, ISEQ is a temporary array used by subroutines ORDER and REORDER.

The technique used in the sort is as follows:

- Read a record from LUNO which consists of the sortword and packed print values. If a complex, the record contains the lead element and all elements within the complex.

- If the current sortword is greater than the last sortword placed on file LUNT, information is transferred immediately to LUNT.

- If the current sortword is less than the last sortword placed on file LUNT, working arrays PL, ISORTA, and LPTR are updated.

- Another record is read from LUNO and processing continues.

When any of the working arrays is filled, reading of LUNO temporarily stops. Array ISORTRA is sorted by subroutine ORDER with the sorted indexes placed in array ISEQ and array LPTR is sorted by subroutine REORDER. The contents of array PL is now merged with file LUNT and results written onto file LUNN. File LUNT is rewound, working arrays cleared, and processing continues by reading input LUNO. Subsequent merges of working arrays with information on LUNT will place groupings of sorted data onto LUNN.

When input LUNO reaches end-of-file, the values of LUNO and LUNN are swapped and the entire process begins again. This cycling, of LUNO and LUNT, is repeated until the end-of-file on LUNO is reached before any of the working arrays have been filled. The SAMPLE TARGET is now printed from file LUNO.

Subroutine MAKEPRFL is illustrated by figure 62.3.

WORD NUMBER

Word Number	Field	Bits
1	0	1
	Region	2
	CC1*	5
	0	1
	CC2*	5
	NDESIG (Alpha)	12
	NDESIG (Num.)	10
2	NCLASS	36
3	NTYPE	36
4	NAME	36
5	ICK	2
	NT(51)	13
	TGTLAT**	21
6	INDEXNO	14
	TGTLON**	22
7	0	9
	MULT	3
	VTO**	24

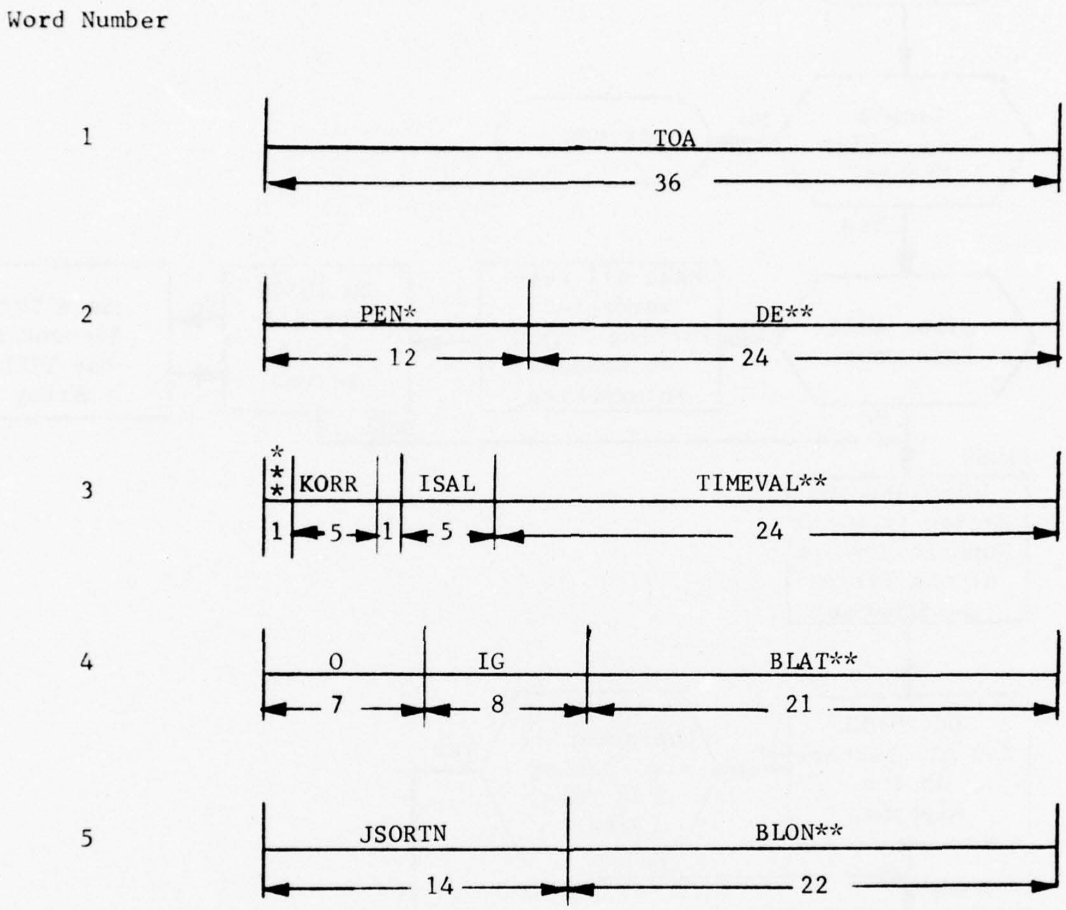
Figure 62.1 Location of Packed Target Values for the Sample Target List (Part 1 of 2)

Word	Field	Field	Field
8	0	NUM	VTOC**
	7	5	24
9	ICOMP	0	VULN
	6	6	24

*CC1 and CC2 are the codes for the 1st and 2nd characters of Country Code

**C in units of .0001.

Figure 62.1 (Part 2 of 2)



* in units of 1/4096

** in units of .0001

*** = 1 if PEN = 1.0

Figure 62.2 Location of Packed Weapon Values for the Sample Target List

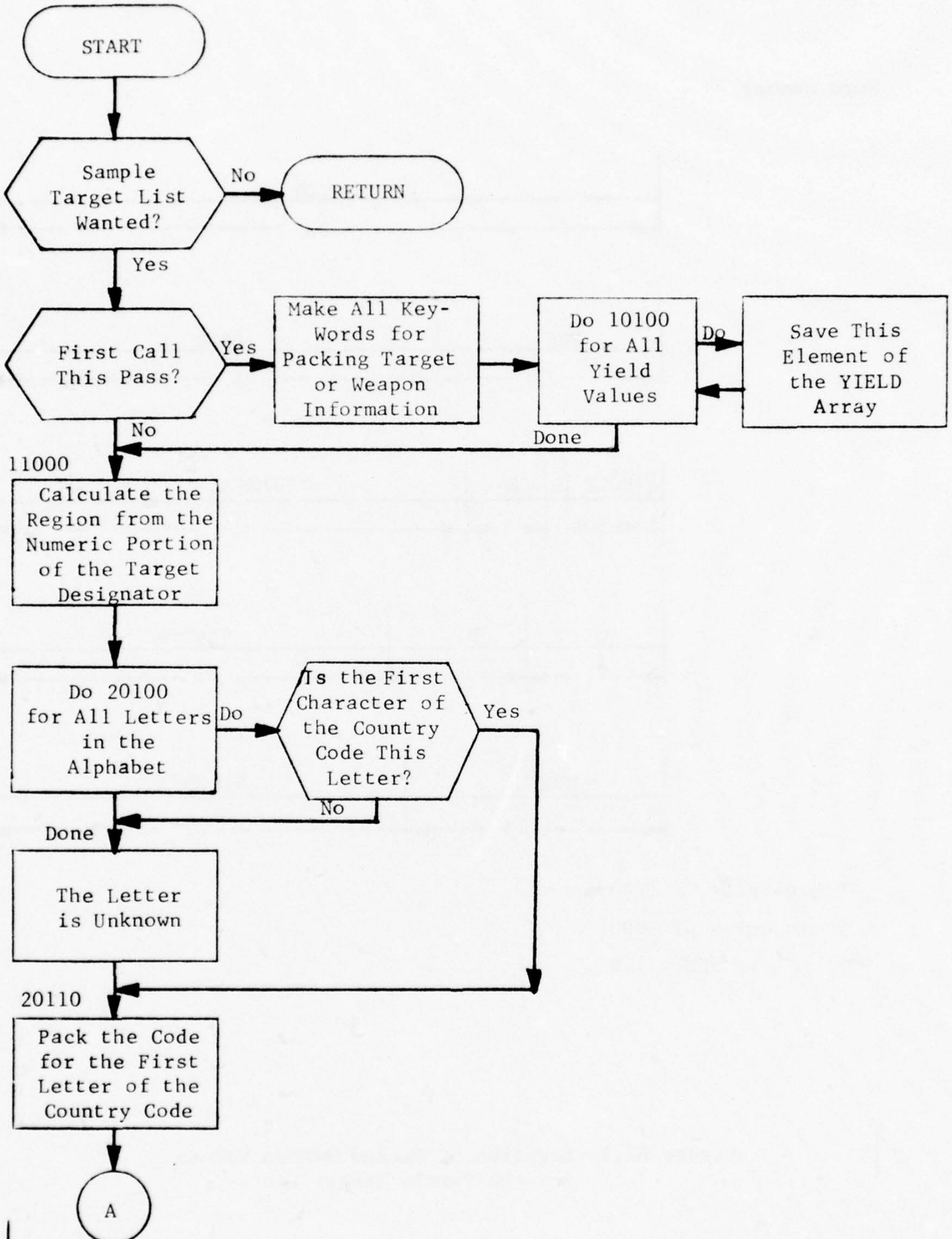


Figure 62.3. Subroutine MAKEPRFL (Part 1 of 7)

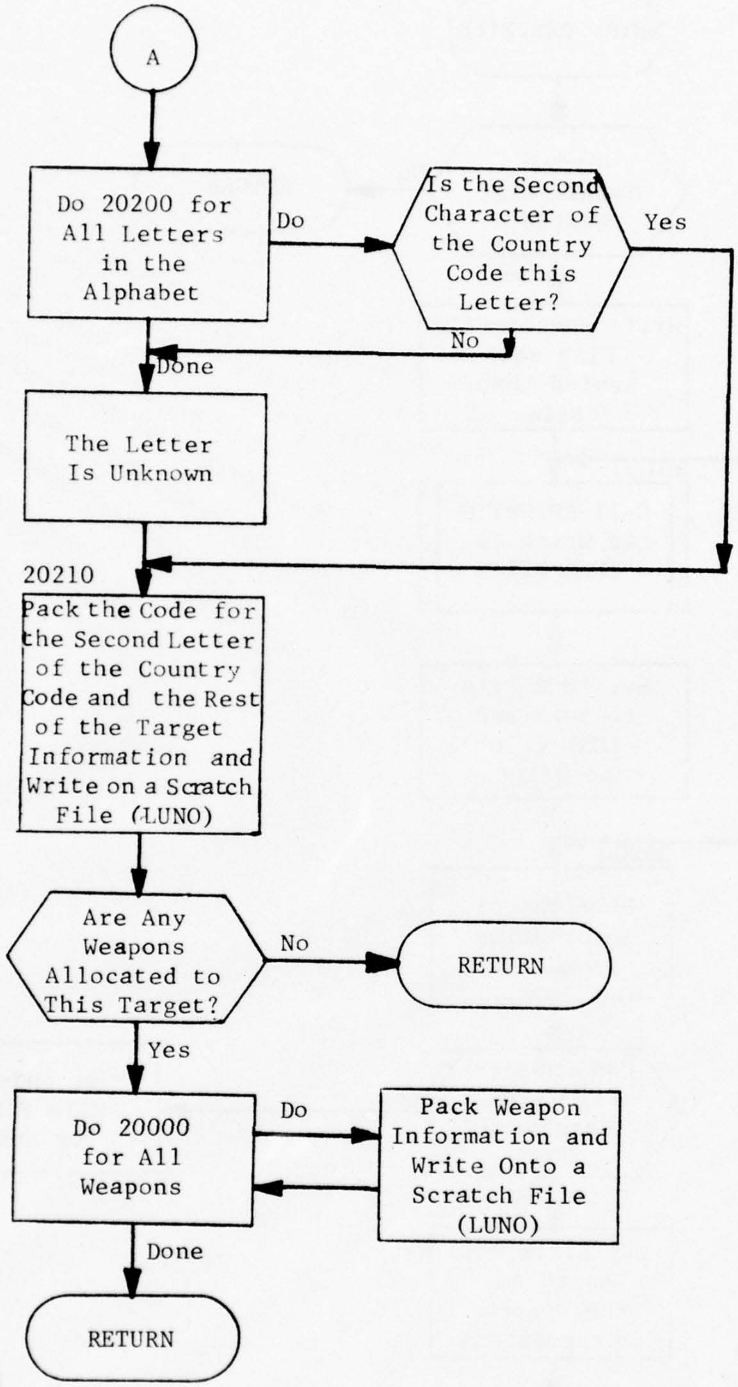


Figure 62.3. (Part 2 of 7)

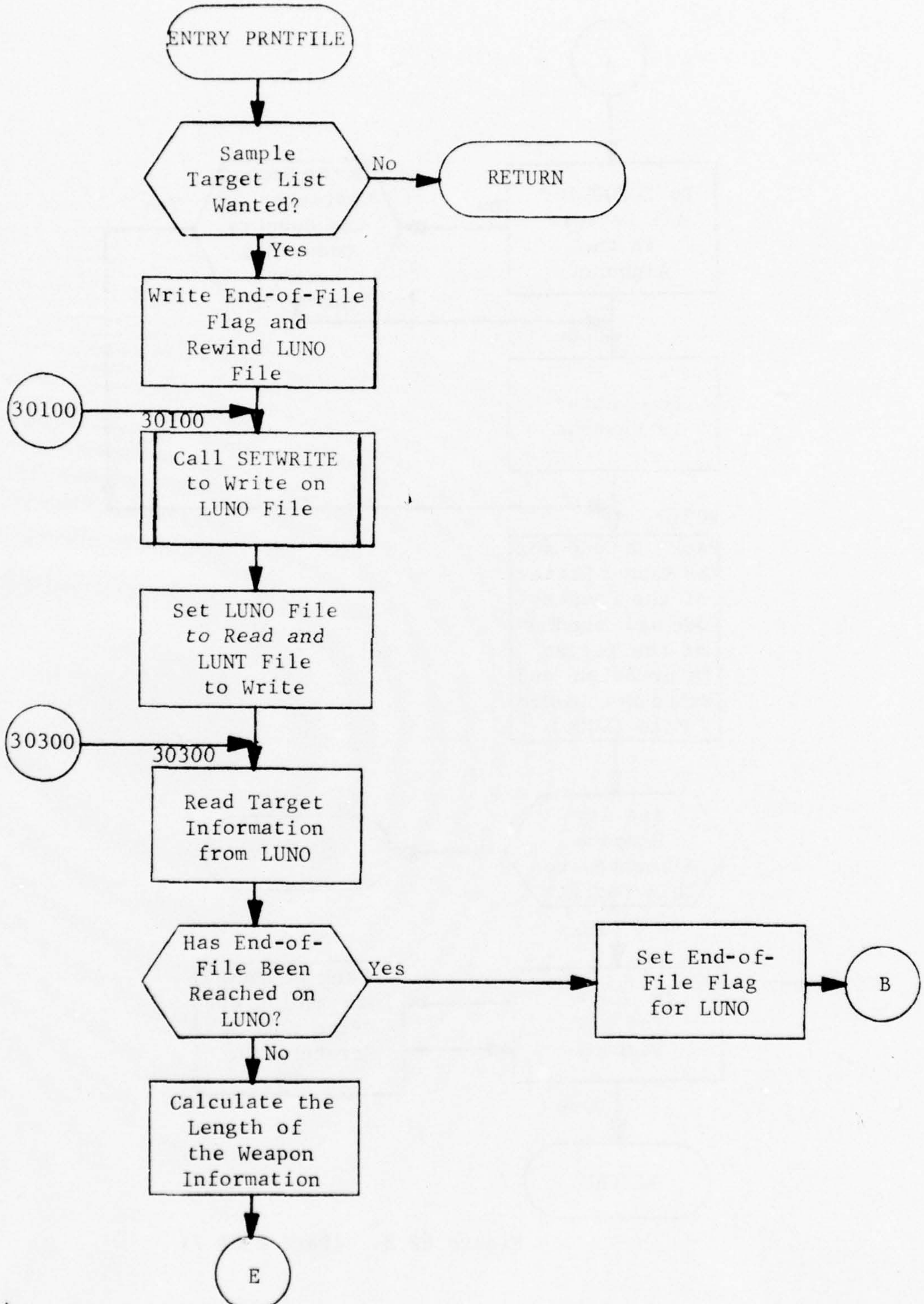


Figure 62.3. (Part 3 of 7)

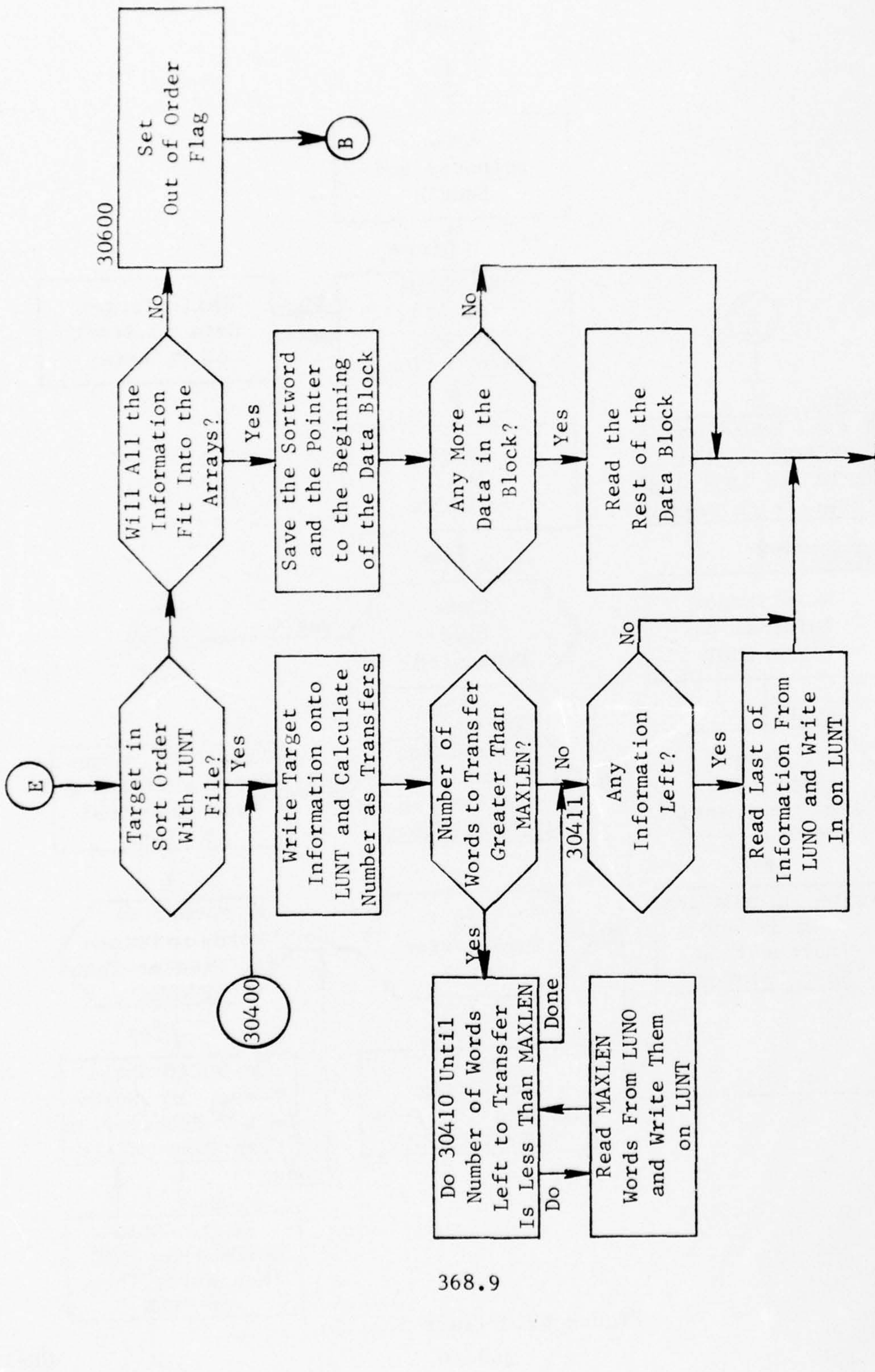


Figure 62.3 (Part 4 of 7)

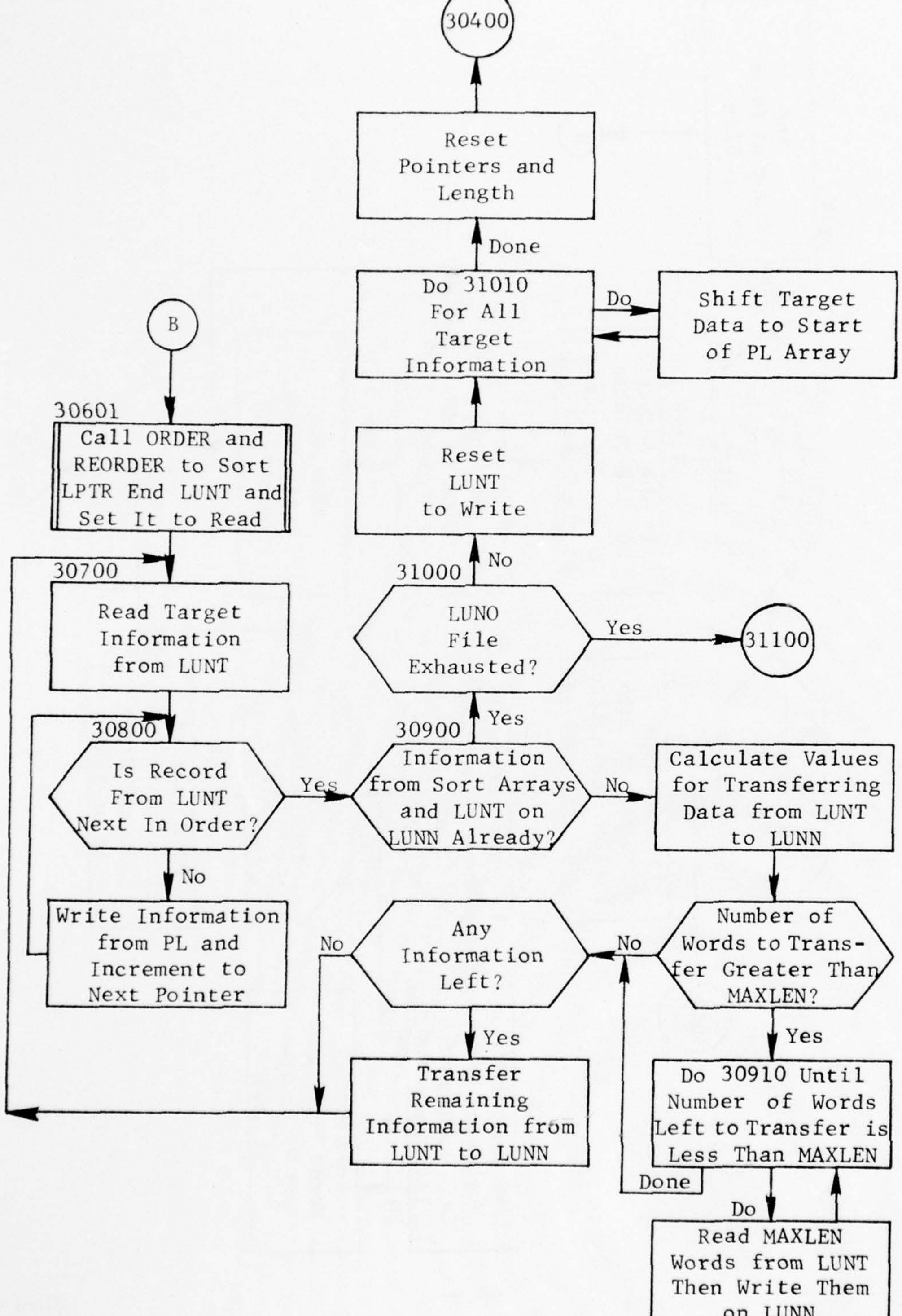


Figure 62.3 (Part 5 of 7)

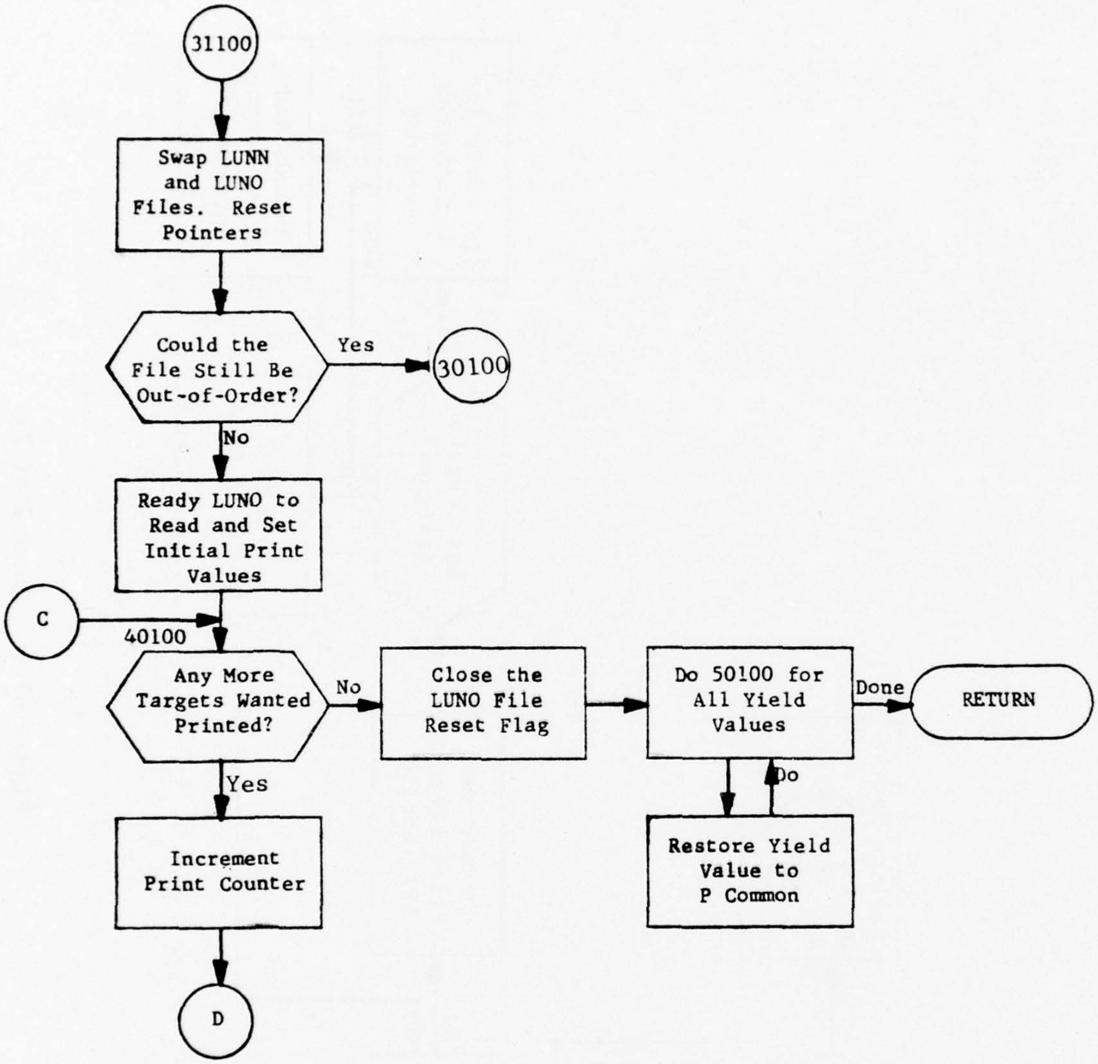


Figure 62.3. (Part 6 of 7)

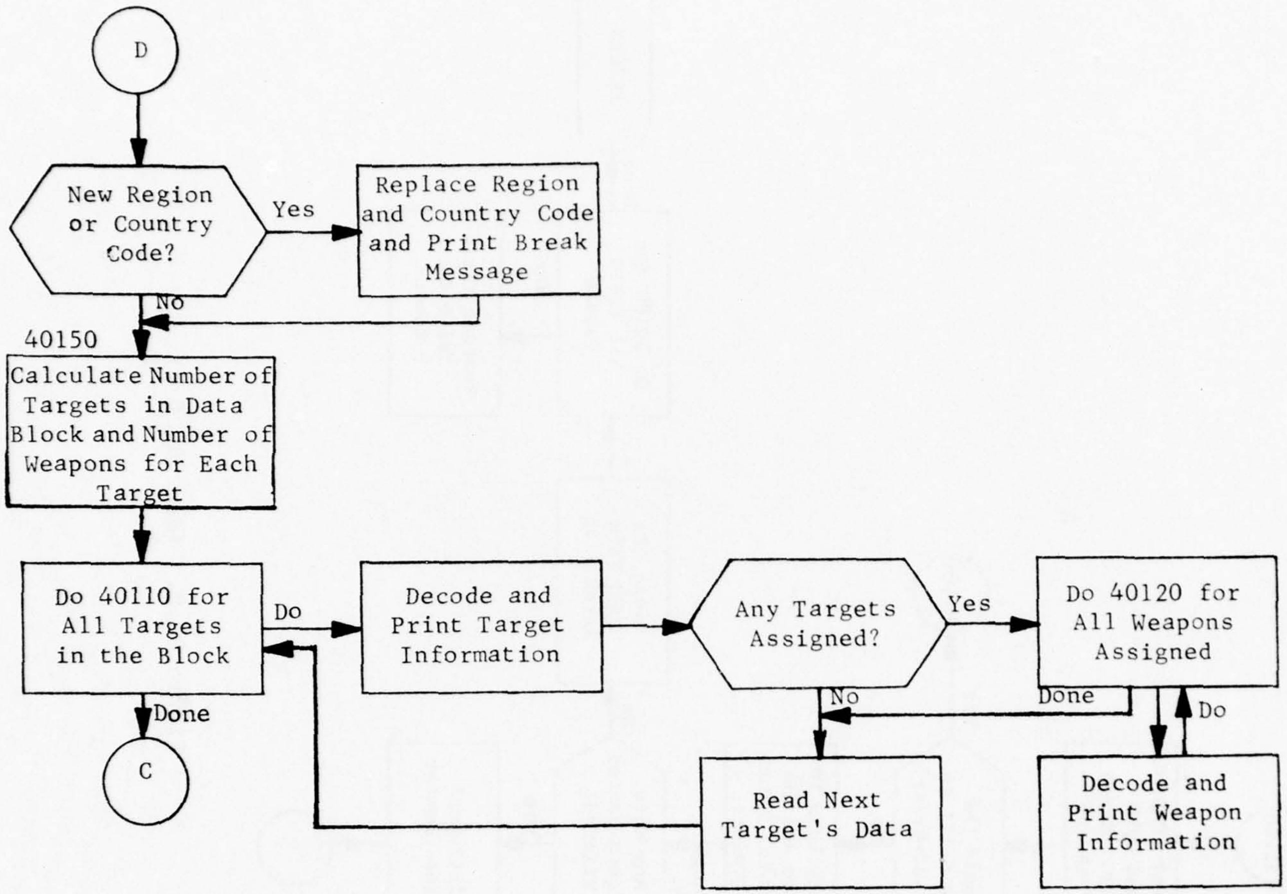


Figure 62.3. (Part 7 of 7)

4.9.B Subroutine MKTARTAB

PURPOSE: To print the TARGET DESIGNATOR/NUMBER DIRECTORY

ENTRY POINTS: MKTARTAB, PRTARTAB

FORMAL PARAMETERS: None

COMMON BLOCKS: BINSCH, CPF, CTGTMOD, ITP, MYIDENT, P, TWORD, WAROUT

SUBROUTINES CALLED: IGET, IPUT, KEYMAKE, ORDER, PAGESKP, RDARRAY, RDWORD, REORDER, SETREAD, SETWRITE, TERMTAPE, WRARRAY

CALLED BY: EVAL2

Method:

MKTARTAB creates the TARGET DESIGNATOR/NUMBER DIRECTORY, and entry PRTARTAB prints the created directory. This directory is a print of sorted target DESIG's with corresponding columns of the target number and if the target is a complex, a third print is given defining the DESIG's of all elements within the complex.

MKTARTAB is called from EVAL2 for each target that is being evaluated during the first pass to write print information on file LUNX. Target designator, target number, and complex status are defined for each record as given in figure 62.4. If the current record defines a simple target, then the designator, number, and a zero are saved on LUNX. The zero flags this record as being a simple target. If the record is the representative element of a complex, ICOMP, the number of elements, will define the third word. For all following elements of a complex, the target designator is placed at the end of this list. If this is a simple target, the number, DESIG, and a zero are saved on a scratch file defined by LUNX. The zero flags this record as being a simple target.

If the target is the representative element of a complex, the number of elements in the complex, ICOMP, will replace the zero. The Target Designator will follow this number.

For all following elements of the third word, the compressed Target Designator is placed at the end of this list.

When the last element of the complex is reached, the entire list is put out for each element of the complex.

This file will be sorted when PRTARTAB is called. It will use the same scratch files that are used in PRNTFILE: LUNN, LUNT, and LUNO. LUNO will be exchanged with LUNX before the sort.

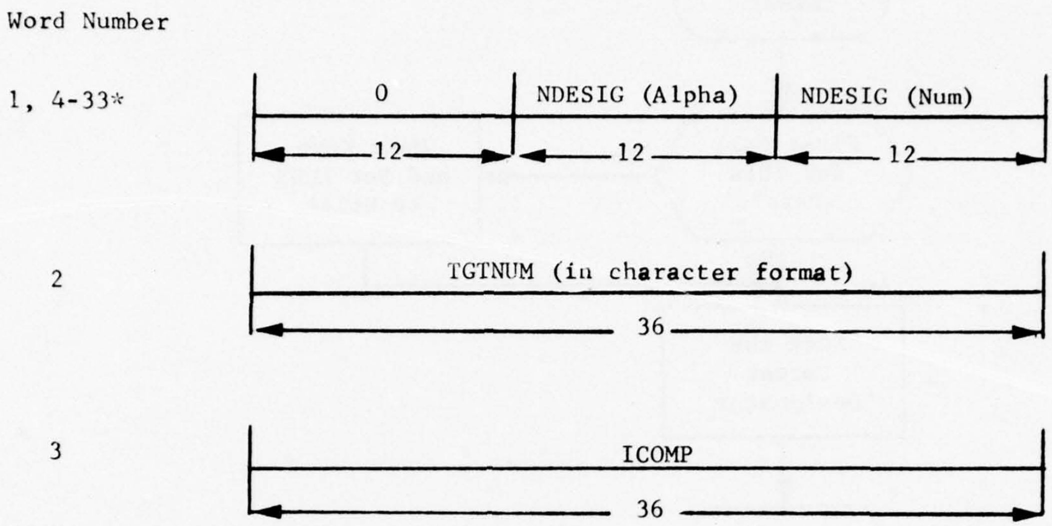
PRTARTAB is called from EVAL2 after all the other tables have been printed out in the first pass. After LUNO and LUNX have been swapped, so that the file with the SAMPLE TARGET LIST information won't be used, the file is sorted in the same manner as in PRNTFILE. It is now ready to be read to print out the Directory.

Three words are read in from the sorted file: the Target Designator in its compressed form, the Target Number, and the length of the record still to be read. For Simple Targets this length is zero.

The Target Designator and Target Number are entered into the arrays from which they will be printed when the page is full. If it is a simple target, the third column is blank. If the target is a complex, the elements of the complex are read in and placed in the third column. The first two columns for the nonrepresentative elements are blank.

When all the information will not fit into a set of columns, the information is put in the next set of columns on the page. When four sets of columns are completed the page is full. It is printed out and the arrays for the new page are begun. Any unused portions of the page are filled with blanks.

MKTARTAB is illustrated in figure 62.5.



*Note: Words 4-33 are not used for simple targets. For a complex, word 4 is the Representative Target and words 5 to ICOMP+3 are subelements of the complex.

Figure 62.4 Location of Packed Values for the TARGET DESIGNATOR/NUMBER DIRECTORY

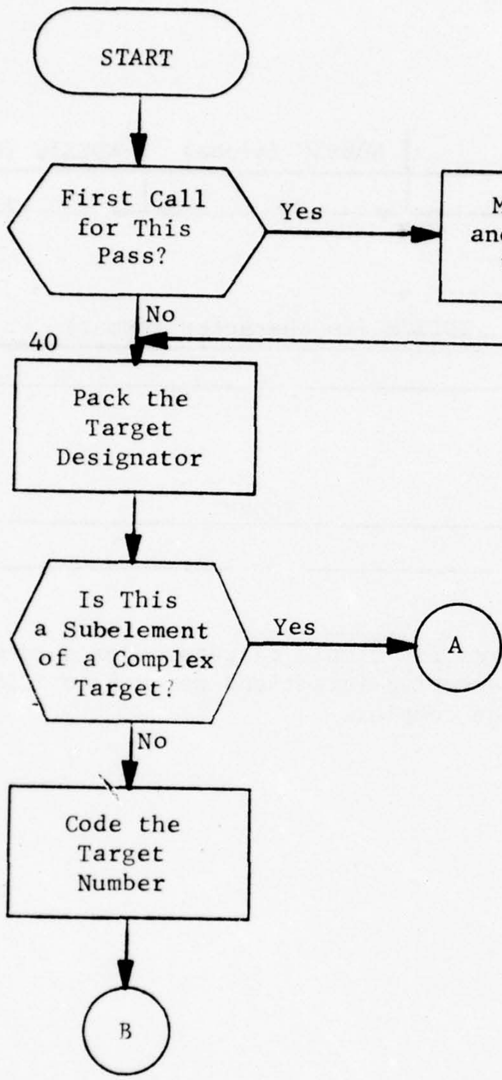


Figure 62.5. Subroutine MKTARTAB (Part 1 of 7)

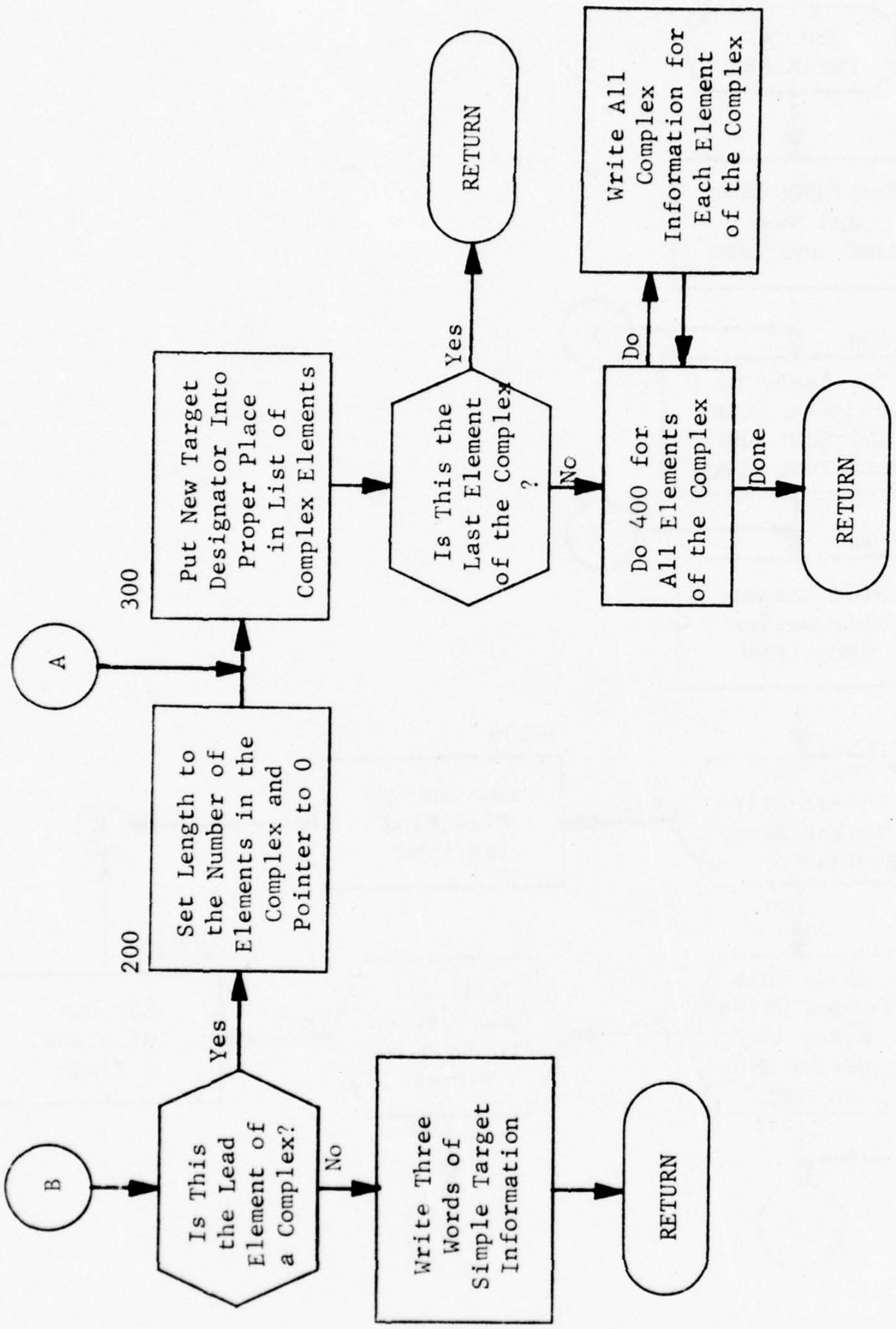


Figure 62.5. (Part 2 of 7)

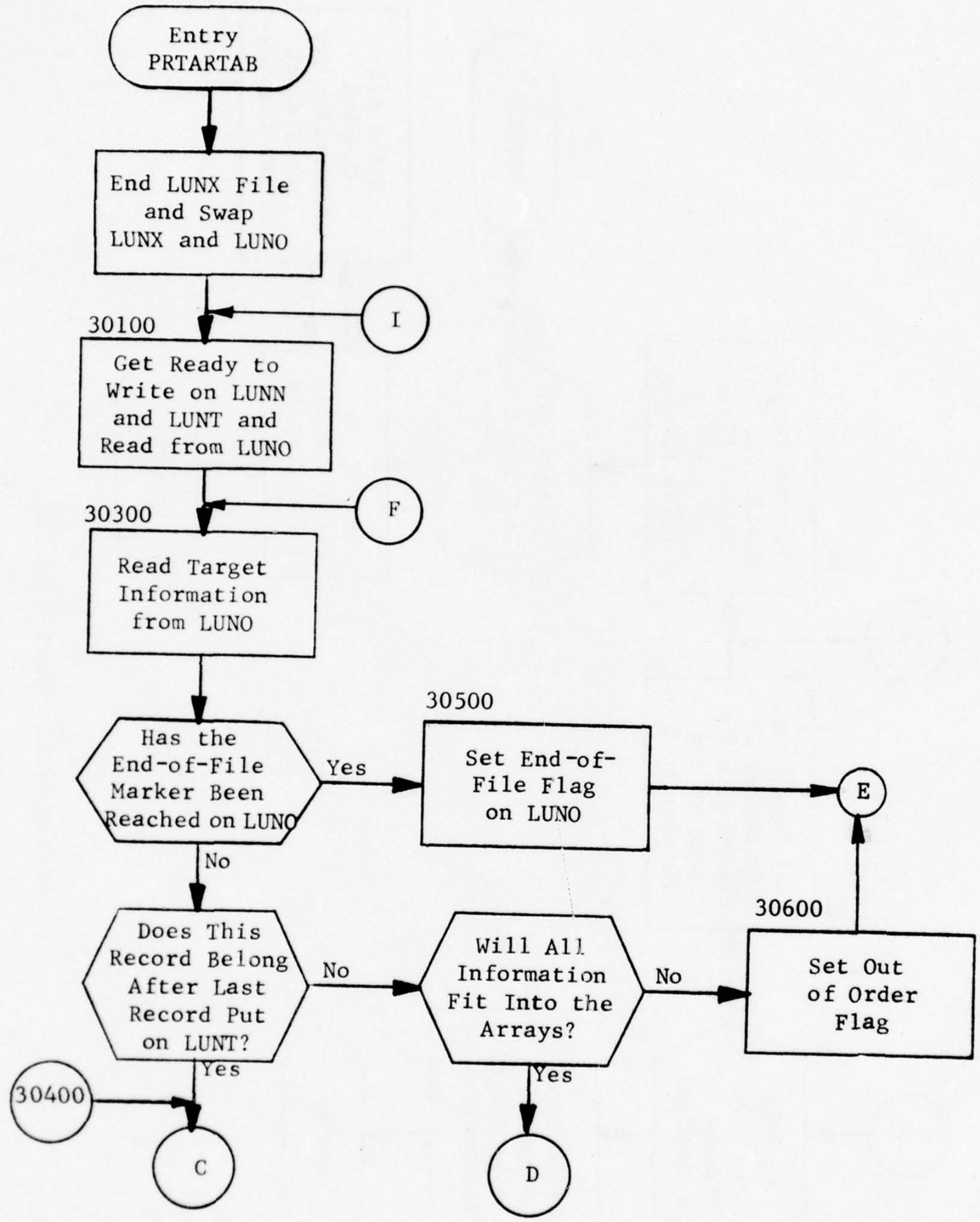


Figure 62.5. (Part 3 of 7)

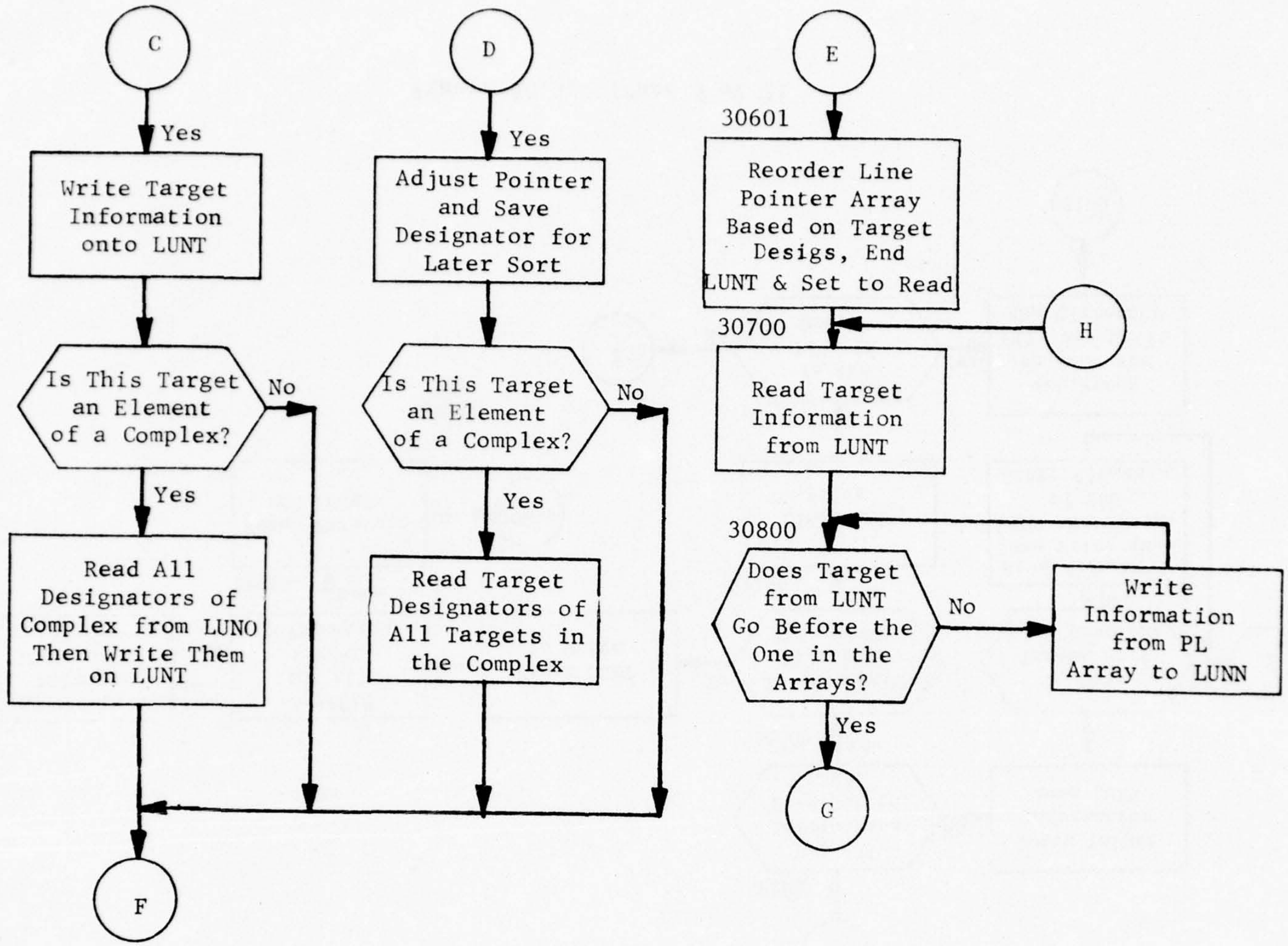


Figure 62.5. (Part 4 of 7)

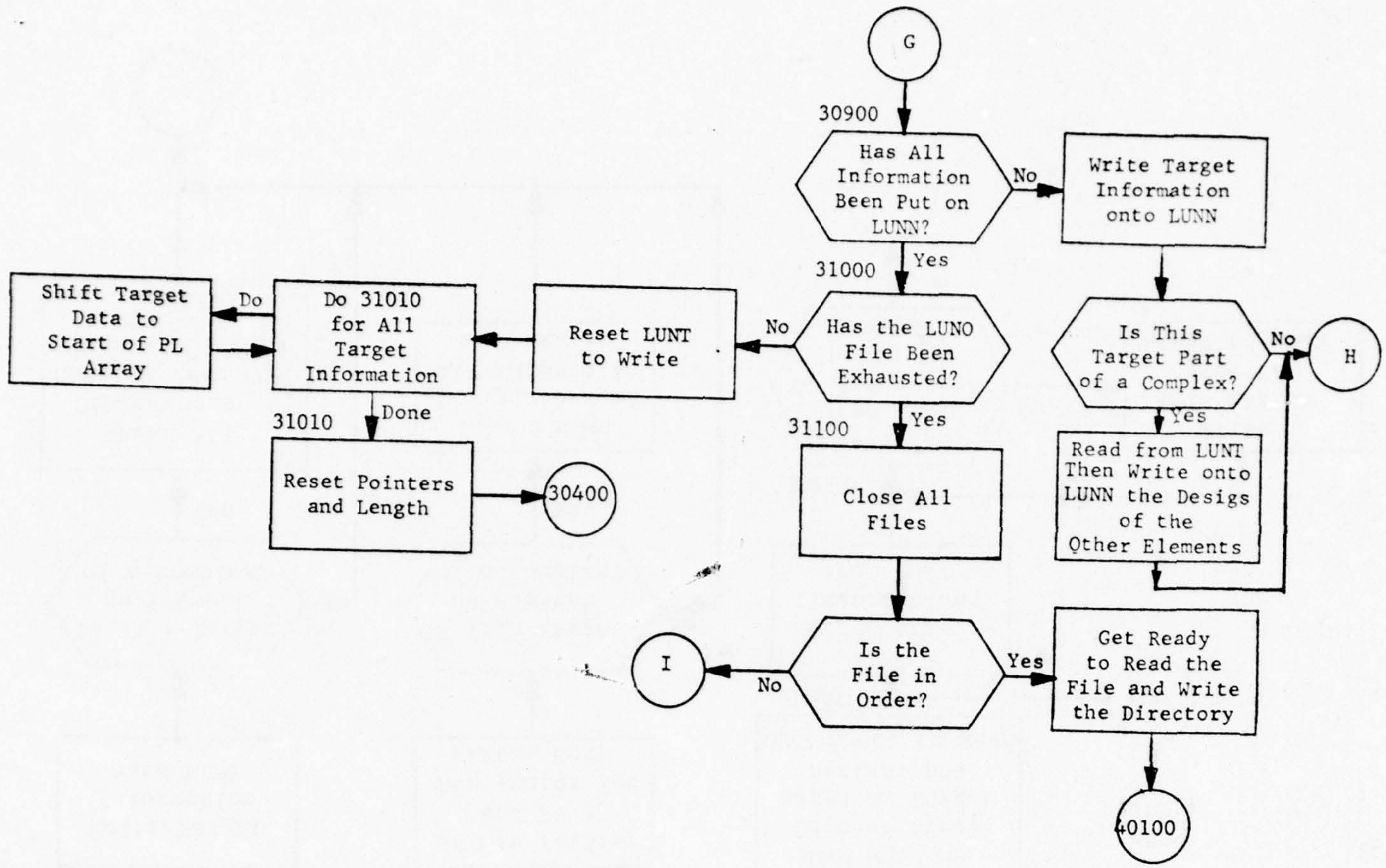


Figure 62.5. (Part 5 of 7)

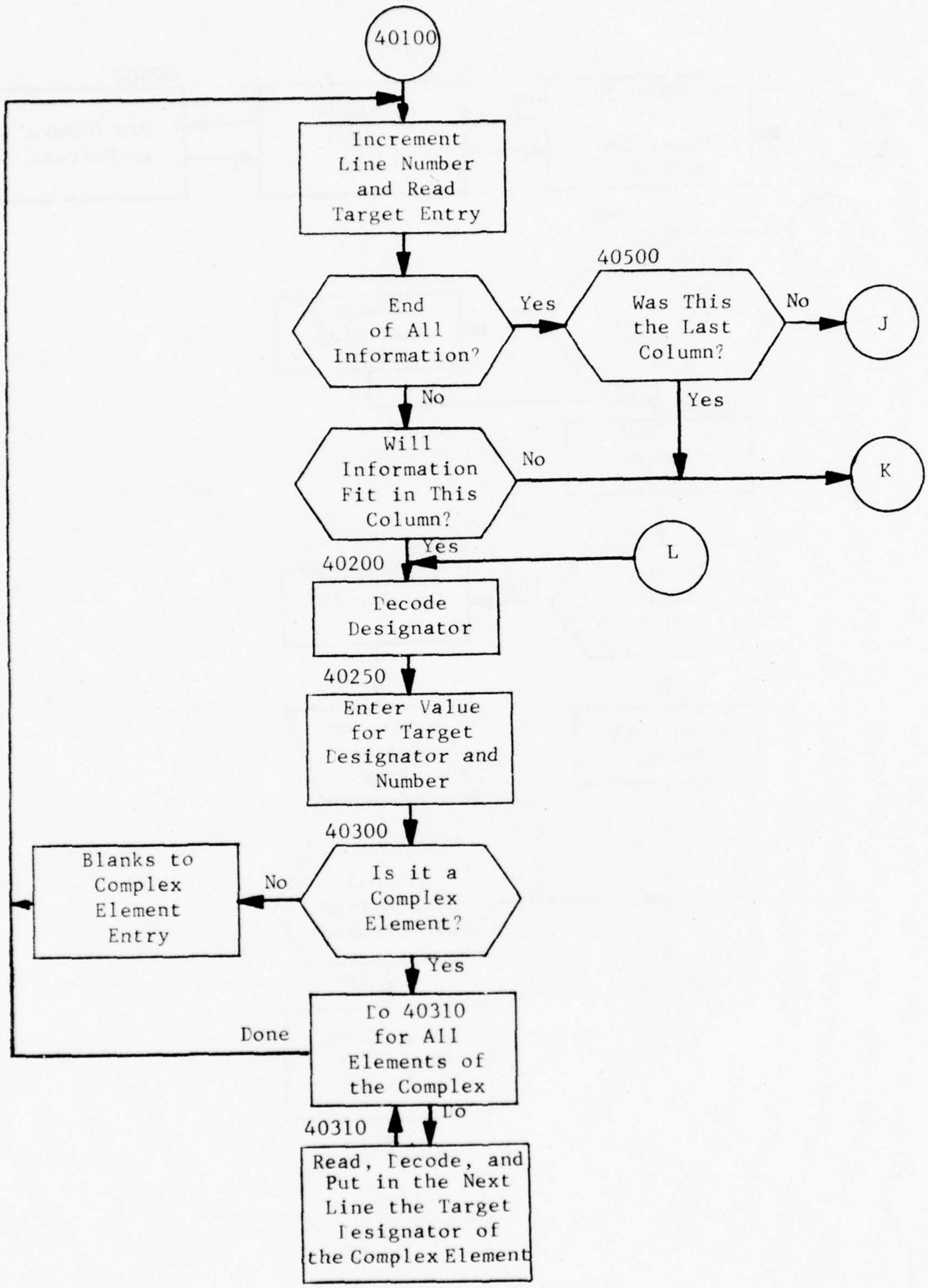


Figure 62.5 (Part 6 of 7)

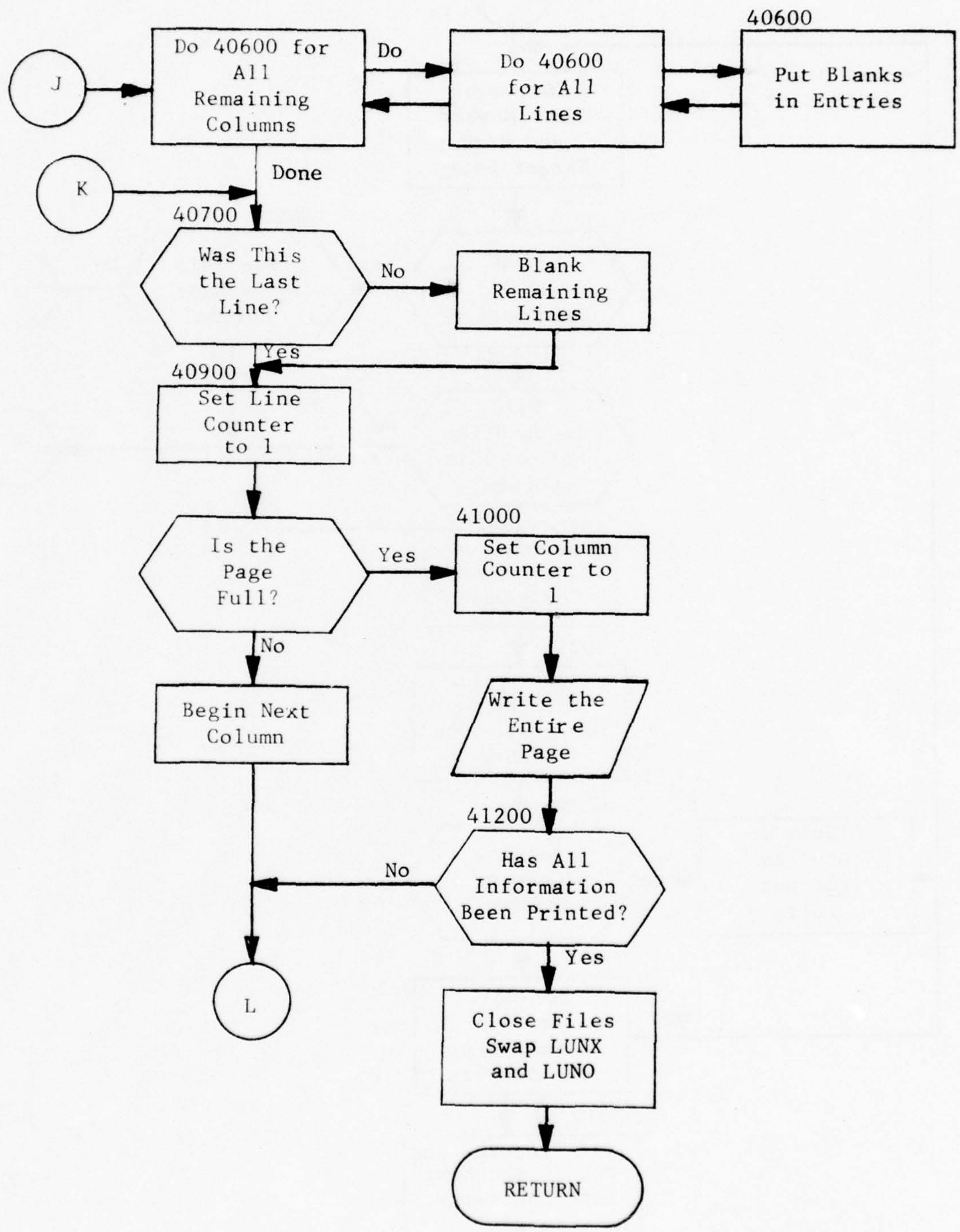


Figure 62.5. (Part 7 of 7)

4.11 Subroutine PACK

PURPOSE: PACK with its associated subroutines BOMRPAKR, MISLPAKR, SEARCH, and UNPACKER converts the sortie-order weapon allocation on the PLANTAPE to target-ordered allocation and writes this new allocation onto a SCRATCH file which is treated like the ALOCTAR file in subsequent processing by EVAL2. PACK is used only in post-PLANOUT operation of EVALALOC.

ENTRY POINTS: PACK

FORMAL PARAMETERS: None

COMMON BLOCKS: BINSCH, CTGTMOD, FILABEL, FILES, HOB, KEY, ISORTN, ITP, LATLON, MYIDENT, MYLABEL, N, NOPRINT, OPT, P, PLANREC, TWORD

SUBROUTINES CALLED: ABORT, BOMRPAKR, IPUT, KEYMAKE, MISLPAKR, ORDER, RDWORD, RDARRAY, REORDER, SETREAD, SETWRITE, SKIP, TERMTAPE, UNPACKER, WRARRAY, WRWORD

CALLED BY: EVALALOC

Method:

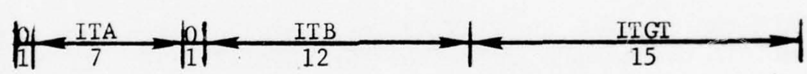
Before PACK can reorder the PLANTAPE weapon allocation into target order, it must obtain the target order from the ALOCTAR file. It does this by reading the target index number and target number one target at a time from the ALOCTAR file and by packing these into the next unfilled word of the JORDER array until the whole file has been read (see figure 64). Then it uses ORDER and REORDER to reorder the array in target index number, INDEXNO order (since the PLANTAPE does not contain target number data). PACK is then ready to process the PLANTAPE, which it does one record at a time. Tanker records on the PLANTAPE are ignored by PACK since they contain no weapon allocation data.

PACK packs data for each event on the PLANTAPE which is a target event. When processing a bomber record, it calls subroutine BOMRPAKR once for each event involving a target burst to pack the weapon allocation data into the next unused words in the JLINK, KLINK, LLINK, and SLINK arrays and to determine the target number on the ALOCTAR file which correspond to this target index number. Subroutine MISLPAKR is called by PACK to perform the analagous function for missile target events. This process is repeated for every missile and bomber record on the PLANTAPE until the JLINK, KLINK, LLINK, and SLINK arrays are filled. Then PACK reorders these arrays in target number order. If the arrays are filled for the first time, they are simply written onto a SCRATCH file in the new order. If not, they are merged with the packed data which must be read

from the previously written SCRATCH file; the merged data, which is in target number order, is written onto a new SCRATCH file. Then PACK continues reading and processing PLANTAPE records as before until the packed arrays are again filled or until the PLANTAPE has been completely read and all the weapon allocation data are contained in target number order on one scratch file. (If the total number of warheads is no greater than 5,000, no SCRATCH file is needed or written.)

The final phase of processing by PACK consists of reading the ALOCTAR file one target block at a time, calling UNPACKER to unpack the associated weapon allocation data for the target from the previously written SCRATCH file, and writing the ALOCTAR target data and PLANTAPE weapon allocation data together on a new SCRATCH file. When all targets on the ALOCTAR file and weapon data from the PLANTAPE have been processes in this way, PACK writes the ALOCTAR end-of-file record on the ALOCTAR-like SCRATCH file and returns control to EVALALOC.

Subroutine PACK is illustrated in figure 65.



ITGT = Target number
ITA = INDEXNO/2048
ITB = INDEXNO - ITA*2048
(i.e., INDEXNO = ITA*2048 + ITB)

Figure 64. Format of JORDER Array Element

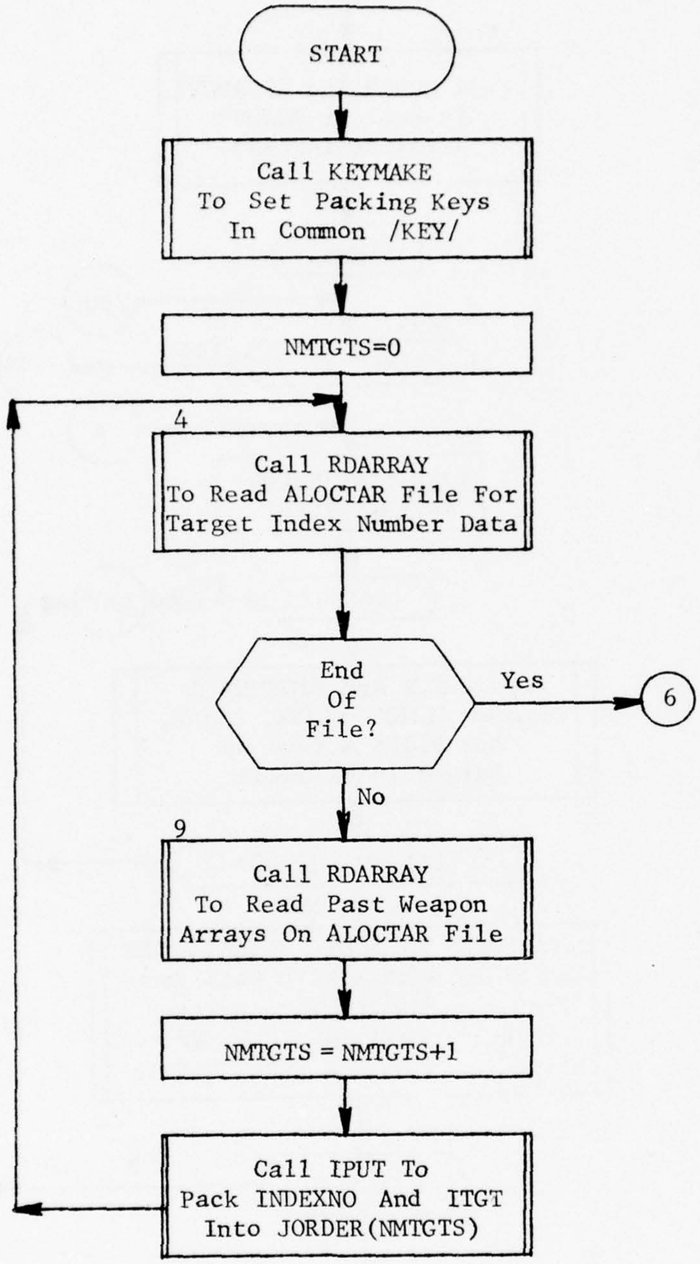


Figure 65. Subroutine PACK
(Part 1 of 4)

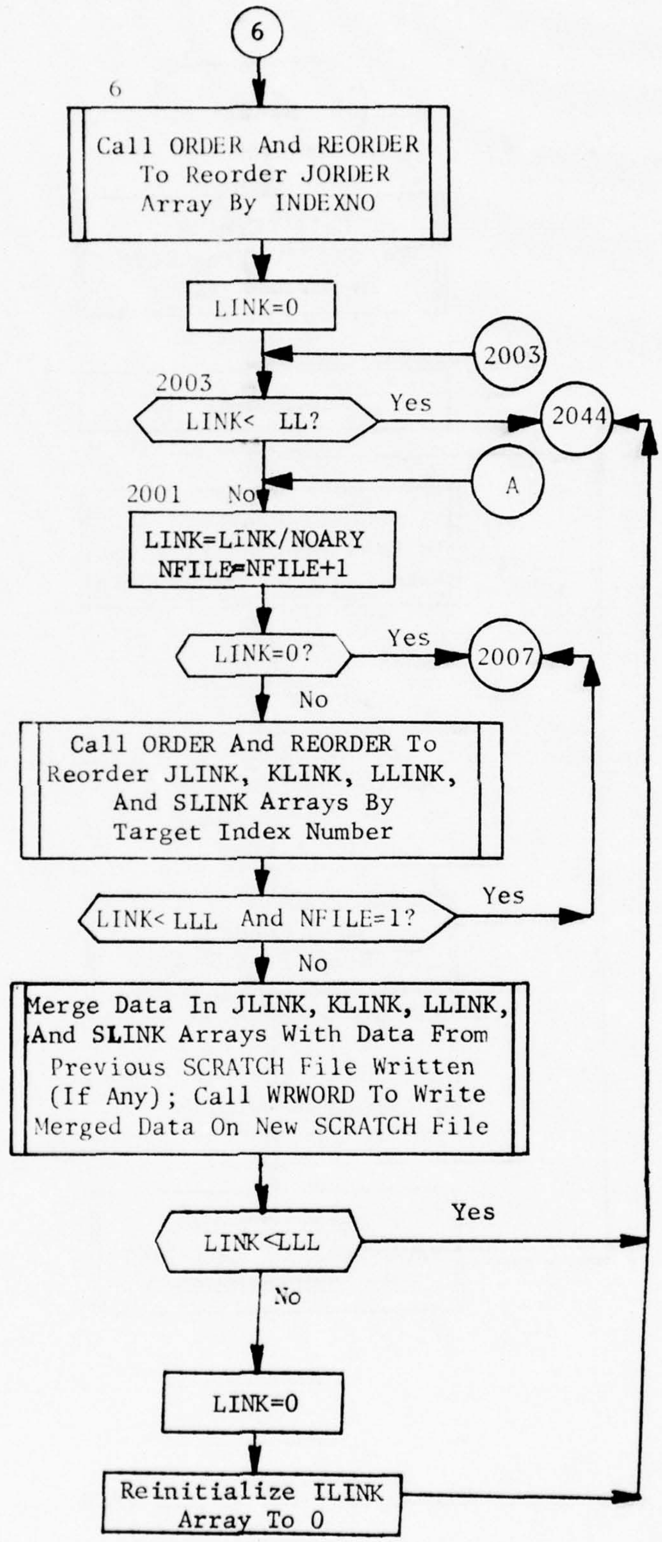


Figure 65. (Part 2 of 4)

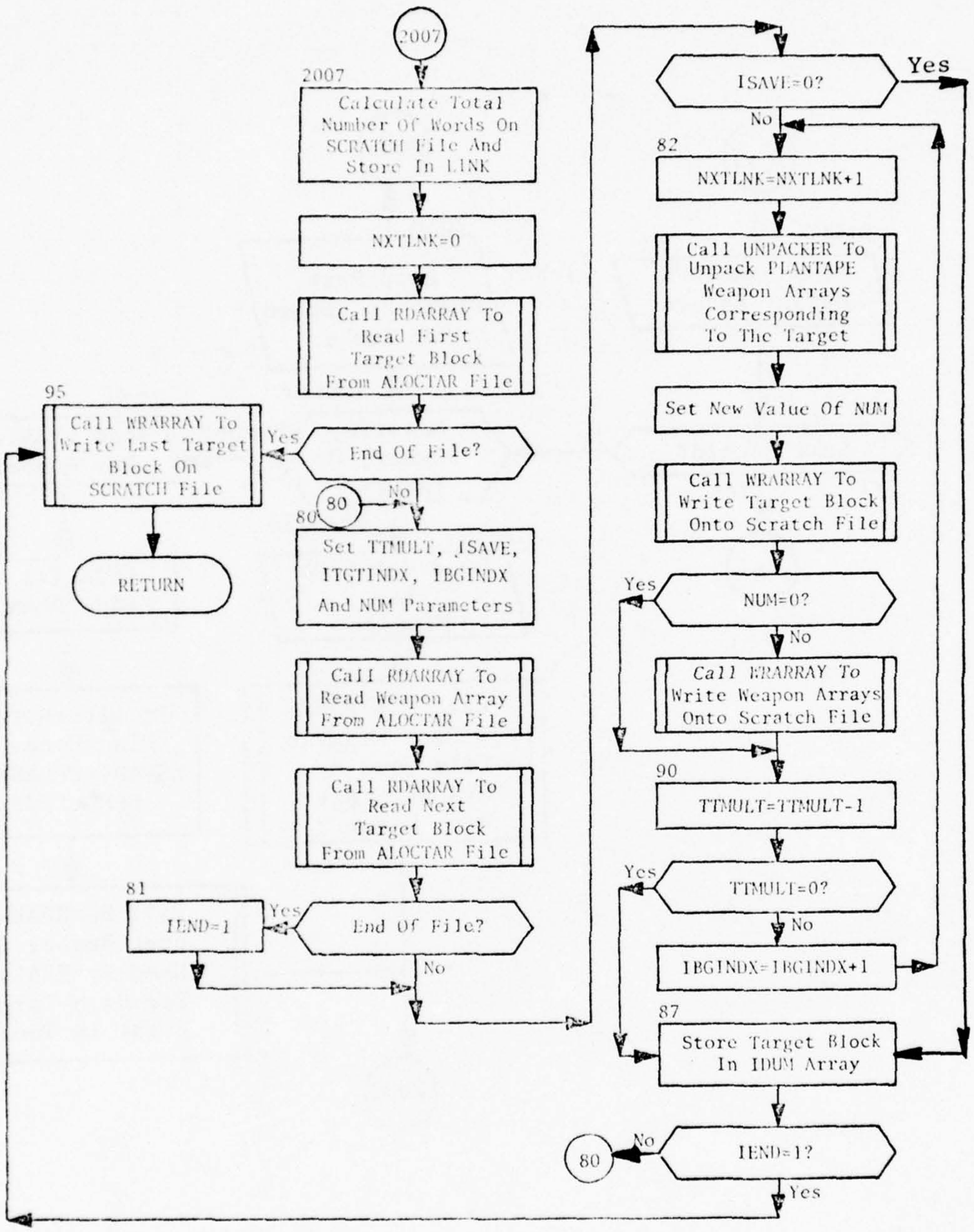


Figure 65. (Part 3 of 4)

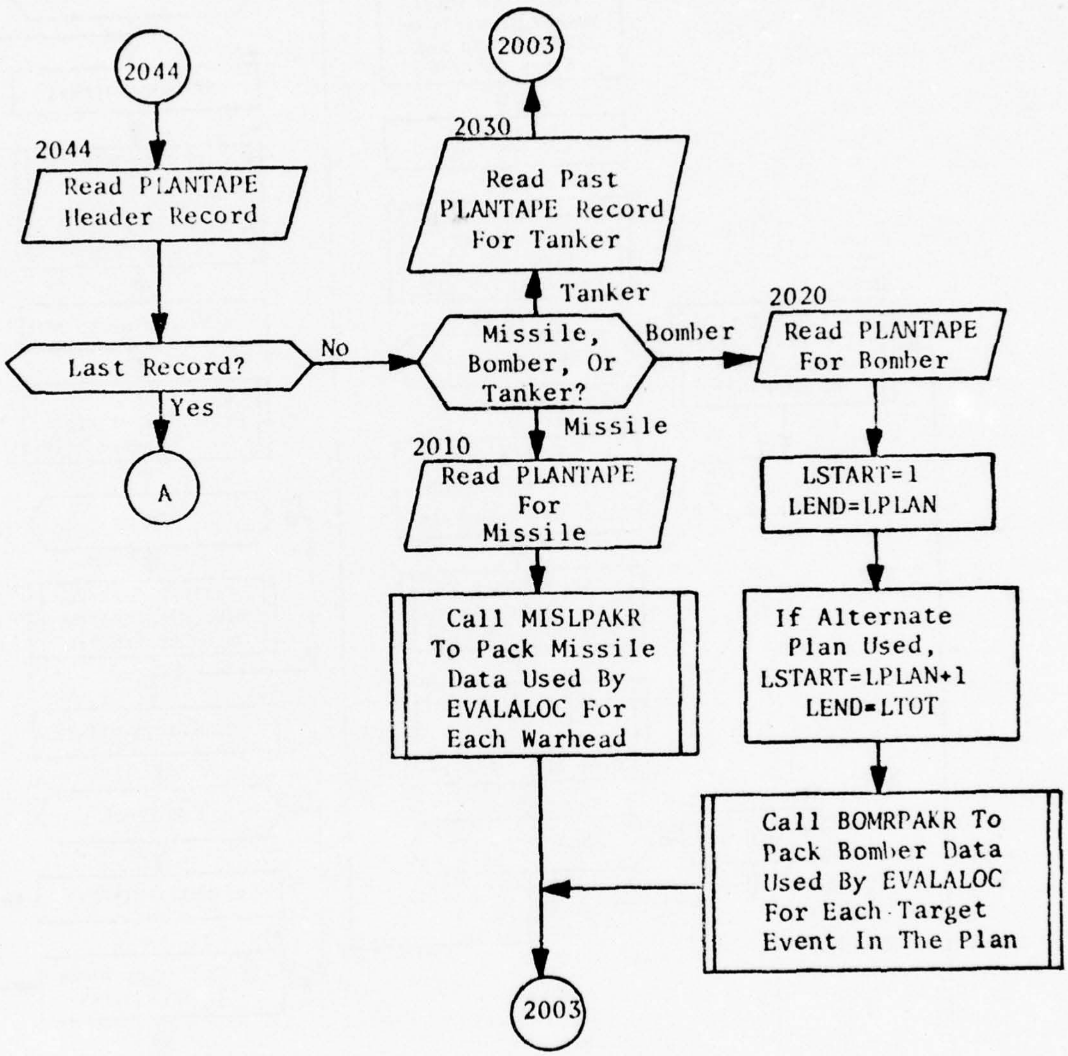


Figure 65. (Part 4 of 4)

THIS PAGE INTENTIONALLY LEFT BLANK

4.12 Subroutine SEARCH

PURPOSE:	To determine the target number from the ALOCTAR file which corresponds to the index number of a PLANTAPE target.
ENTRY POINTS:	SEARCH
FORMAL PARAMETERS:	INDEXNO - The target index number for which the target number is sought ITGT - The target number corresponding to INDEXNO and determined by SEARCH
COMMON BLOCKS:	BINSCH, KEY
SUBROUTINES CALLED:	IGET
CALLED BY:	BOMRPAKR, MISLPAKR

Method:

Subroutine PACK sets up the JORDER array, which is a list of packed words. Each word in the array contains a target index number (INDEX) and a target number (ITGT) corresponding to a target on the ALOCTAR file, and the array is in increasing order by INDEX. SEARCH uses a binary search to find the word in the JORDER array which has target index number INDEXNO or which is the index number closest to INDEXNO but not larget than INDEXNO. This word has index NNOW. SEARCH calls subroutine IGET which unpacks JORDER (NNOW) to obtain ITGT.

The first time SEARCH is called it determines the parameters NLOG, INT, and NOW used in the binary search. These parameters are contained in the common block /BINSCH/.

Subroutine SEARCH is illustrated in figure 66.

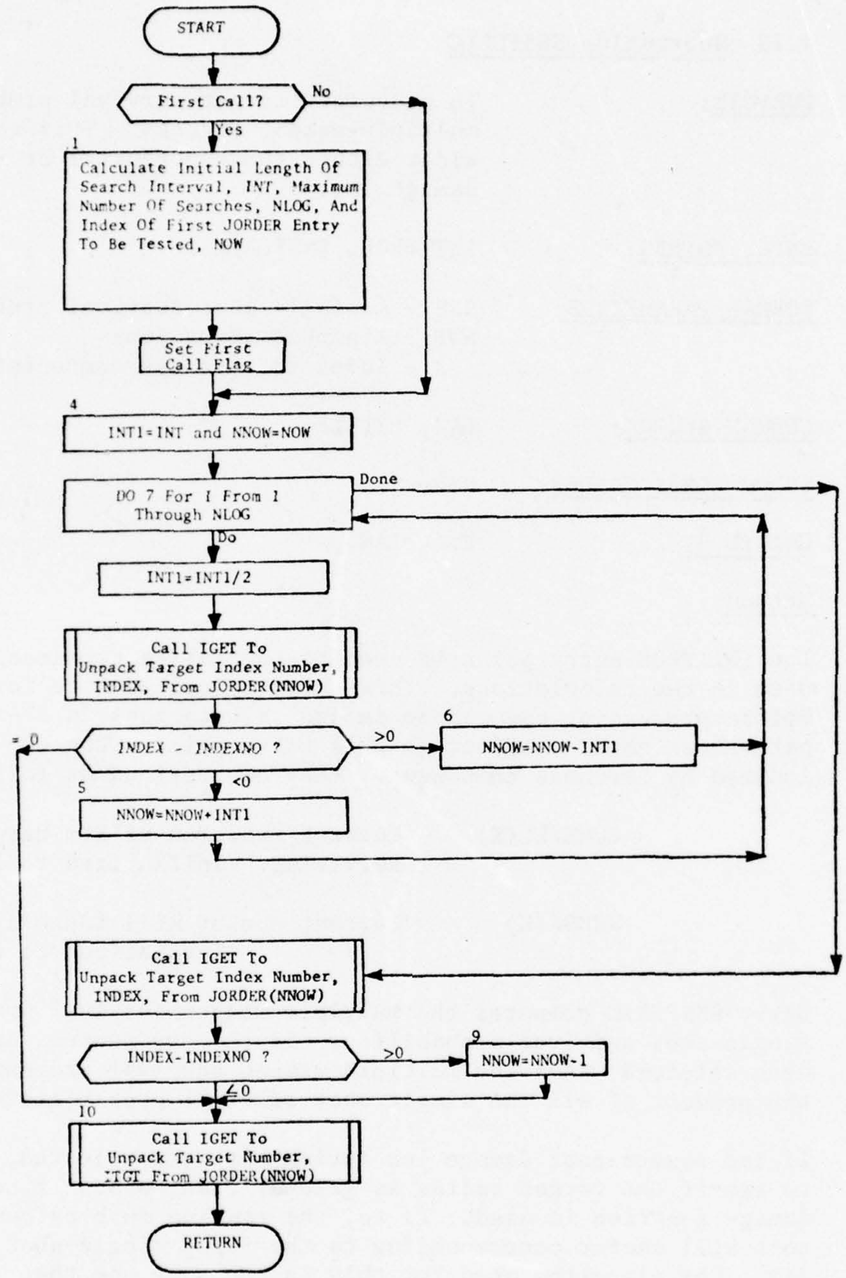


Figure 66. Subroutine SEARCH

4.13 Subroutine SSSPCALC

PURPOSE:	To calculate target survival probabilities for multiple-weapon attacks. This routine will consider either the exponential or square-root damage law.
ENTRY POINTS:	INITPROB, SSSPCALC
FORMAL PARAMETERS:	SSS - A single-shot survival probability NWP - A number of weapons J - Index to hardness component
COMMON BLOCKS:	LAW, LITTLE
SUBROUTINES CALLED:	None
CALLED BY:	EVALPLAN

Method:

The INITPROB entry point is used to initialize two local arrays which are used in the calculations. This entry is called once for each target before processing the weapon damage calculations in EVALPLAN. The formal parameters have no effect on this entry point. The two local arrays are indexed by hardness component. They are defined as follows:

CUMKILL(K)	Current fraction of Kth hardness component surviving. Initialized to 1.0.
SUMSK(K)	Current sum of kill factors for Kth hardness Initialized to 0.0.

Entry SSSPCALC computes the multiple-weapon survival probability from the single-shot survival probability. If the exponential damage option has been selected, then the multiple-weapon survival probability is equal to the product of all the single-shot survival probabilities for each weapon.

If the square-root damage law option has been selected, the routine checks to see if the target radius is greater than zero. If not, the exponential damage function is used. If so, the routine must calculate the square-root kill factor corresponding to the input single-shot survival probability. The algorithm used for this is the same one that is used in subroutine SETABLE in program ALOC. The algorithm is a recursive, one-dimensional search procedure to find the appropriate kill factor. The new kill factors determine a new sum. This new sum defines the new fraction of the target that survives. The multiple-weapon survival probability is then the ratio of the new fraction surviving to the old fraction surviving.

Subroutine SSSPCALC is illustrated in figure 67.

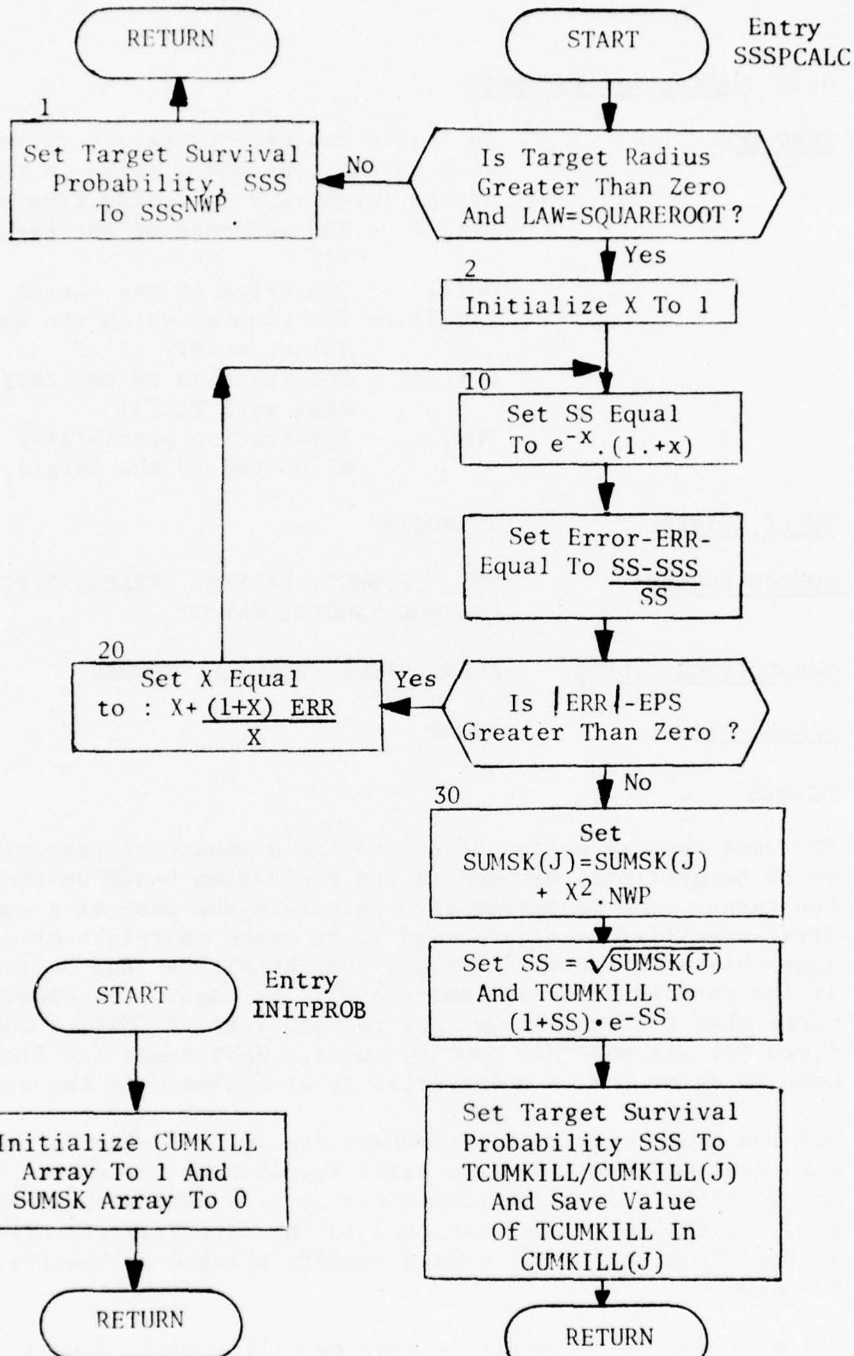


Figure 67. Subroutine SSSPCALC

4.14 Subroutine TGTMODIF

PURPOSE: To enable the user to select or delete targets to be used in the evaluation based on the country code of the target and to modify five target parameters:

- H(1) - The hardness of the target component in PS1
- VO(1) - The value of the target at hardness H(1)
- TAU(1) - The time at which the target changes value rapidly
- FVAL(1) - The fraction of the target value associated with TAU(1)
- PEN - Penetration probability of a weapon allocated to the target.

ENTRY POINTS: TGTMODIF

COMMON BLOCKS: CPF, CWPNMOD, FILABEL, FILES, ITP, MYIDENT, OPT, CTGTMOD, TWORD, WAROUT

SUBROUTINES CALLED: ABORT, ITLE, RDARRAY, RDWORD

CALLED BY: EVAL2

Method:

The user has the option of evaluating a subset of targets. TGTMODIF controls which targets will be used in the evaluation based on the country code of the target, the representative target in the case of a complex. The user first specifies on a data card if he wants to select or delete targets. He does this with the word 'SELECT' or 'DELETE' at the beginning of the card. If the card is blank all targets will be used. The country codes on the cards that follow will use SET to set flags in CCRA, a 26x26 bit matrix of flags for all possible country codes. WANT tests the flag for a country code to determine if a target is to be included in the evaluation.

Modifications of target parameters are made in accordance with target parameter modification data cards supplied by the user. The target type may be ALLTGTS or a specific class or type name such as MILITARY, BEAR, etc. If the type names included in the data base are not known, they can be read from the target damage summary printed on the first pass through EVALALOC.

The attribute to be modified must be H(1), VO(1), TAU(1), FVAL(1), or PEN. If any other word is punched, the program prints an error message (the word punched and a list of allowable words), no changes are made, and execution continues. Modifications made by the subroutine are printed out along with the number of the pass.

The penetration probability PEN is weapon-target dependent. If it is to be modified, the last four words on the data card are the weapon types to which the modification applies. The first of these four words may be ALLWPNS, BOMBERS, or MISSILES if a specific type name is not used. If one of these three general names is used, no other type names will be read from the card.

The target is described by either one or two hardness components. The total value is divided between the hardness components, VO(1) being that part of the target value associated with the first hardness component H(1). If VO(1) is changed, a check is made to ensure that the modification value does not exceed the total target value.

Such a function gives a step-like dependence of the target value on time. Targets such as cities usually have only one time component (NK=1), and TAU(1) is very large. Other targets such as alert bomber bases may have a small value of TAU(1). Usually, the major part of the target value is associated with FVAL(1) (i.e., FVAL(1) is much larger than FVAL(2)).

In all cases TGTMODIF modifies the specified target parameter for the specified target type(s) by multiplying it by the factor XTGTATT which is also specified by the user. If there is more than one hardness or time value component, then both VO(1) and VO(2) or all five of FVAL(1), FVAL(2), etc., will be modified.

Subroutine TGTMODIF is illustrated in figure 68.

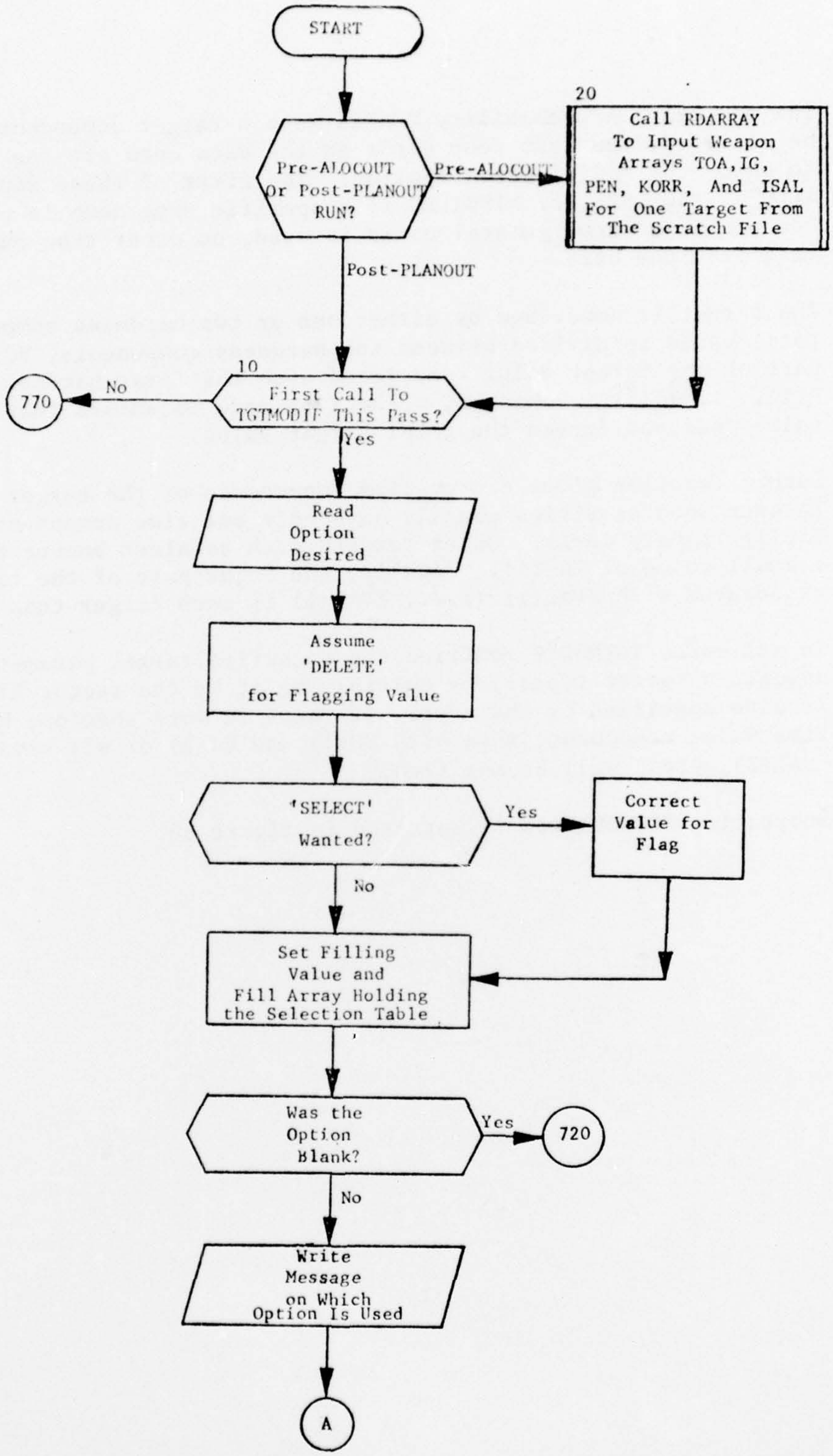


Figure 68. Subroutine TGTMODIF (EVALALOC)
(Part 1 of 5)

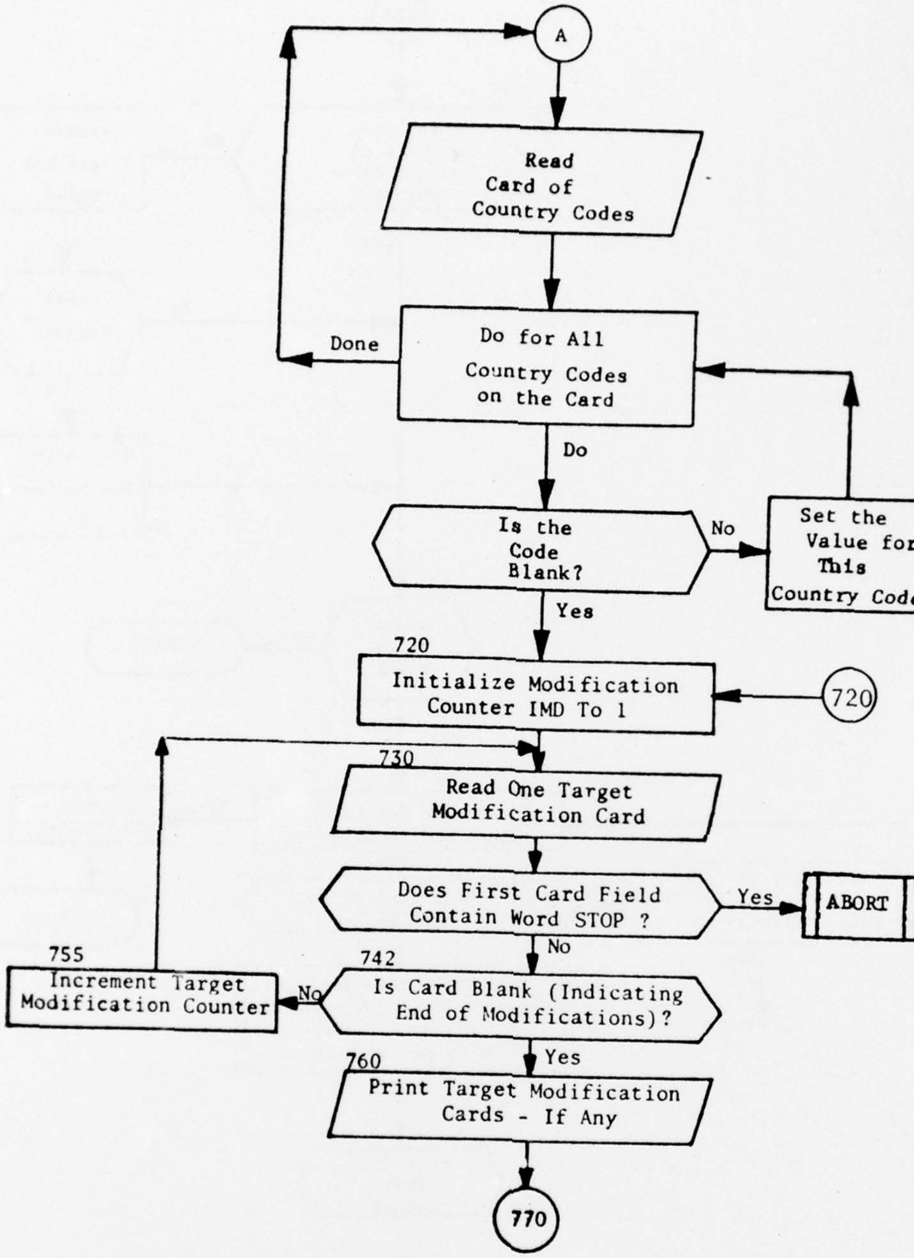


Figure 68. (Part 2 of 5)

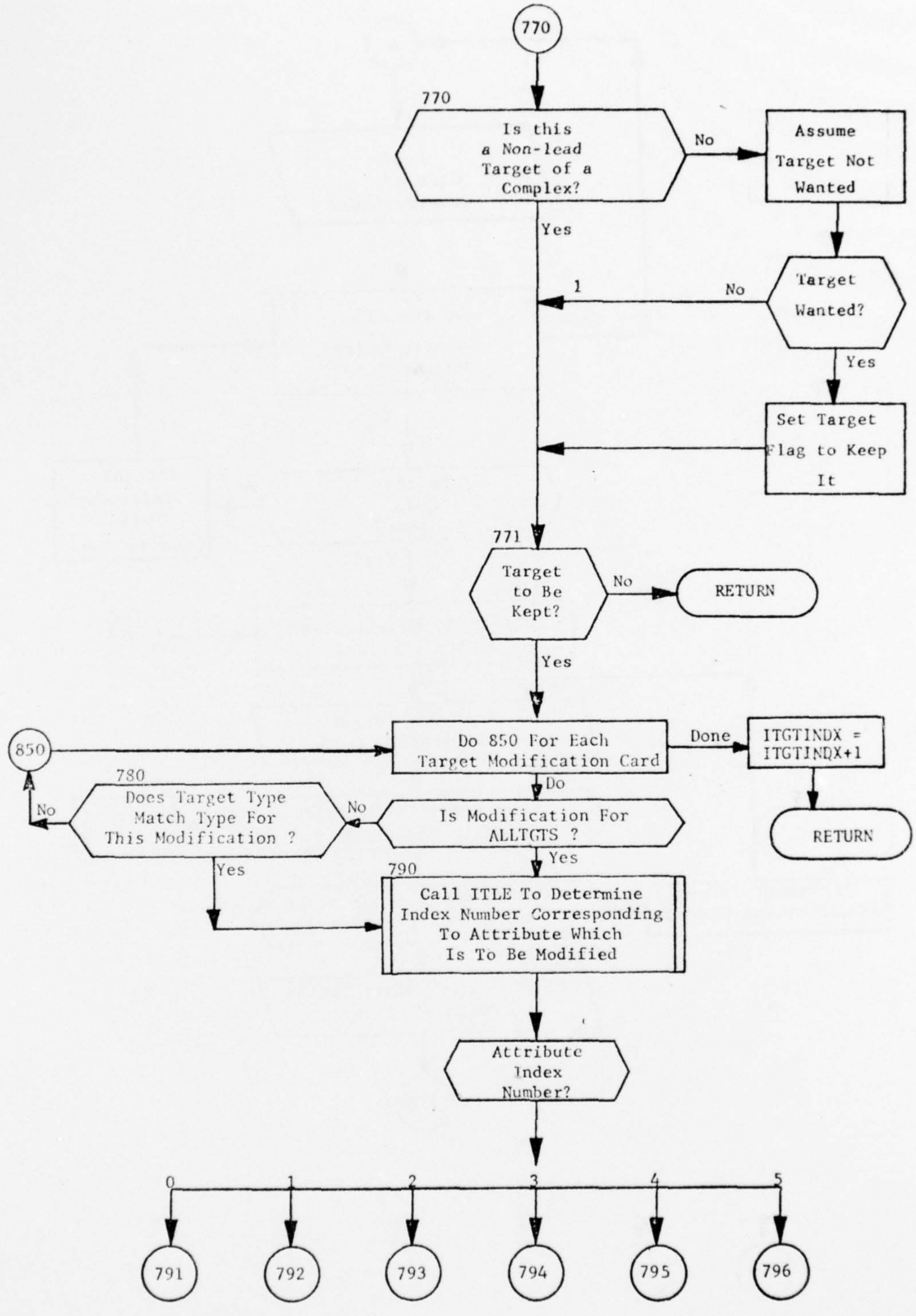
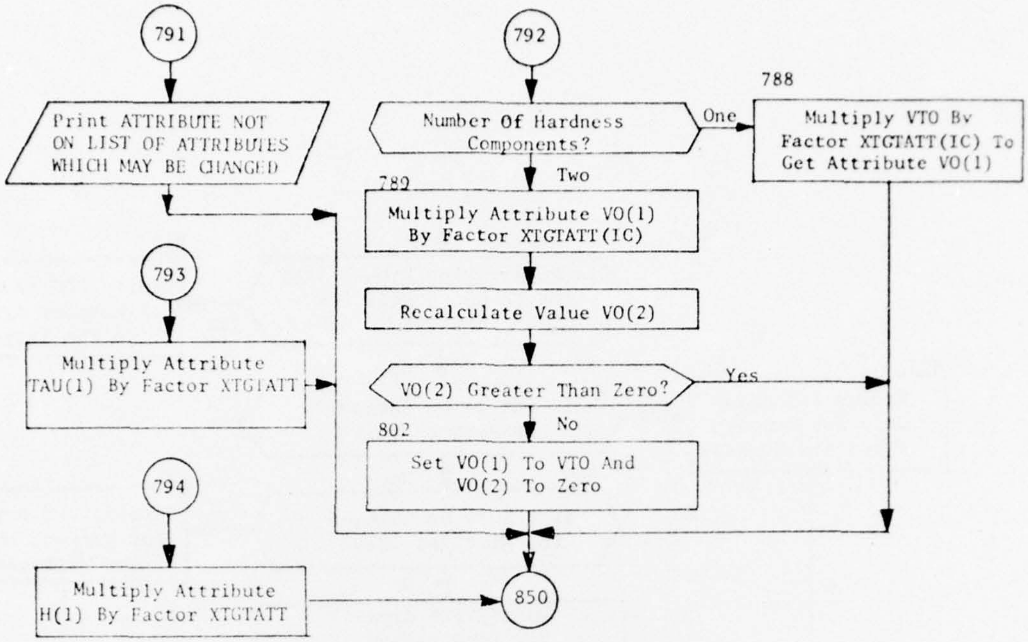


Figure 68. (Part 3 of 5)



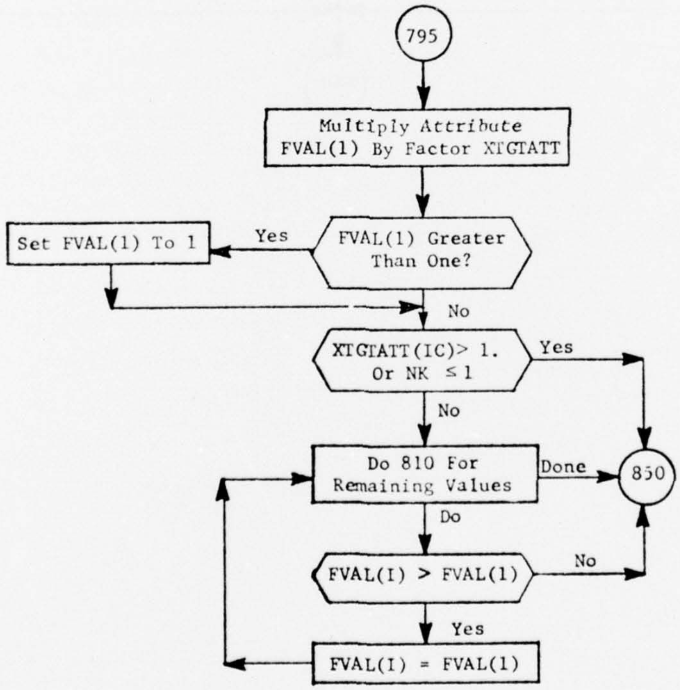


Figure 68. (Part 4 of 5)

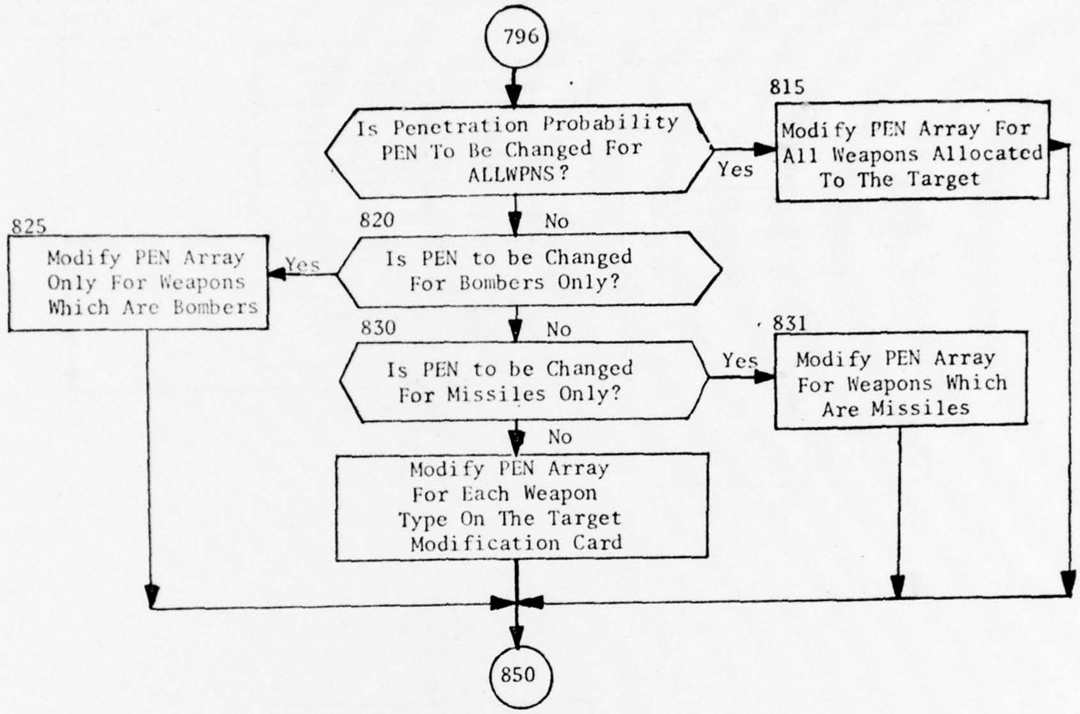


Figure 68. (Part 5 of 5)

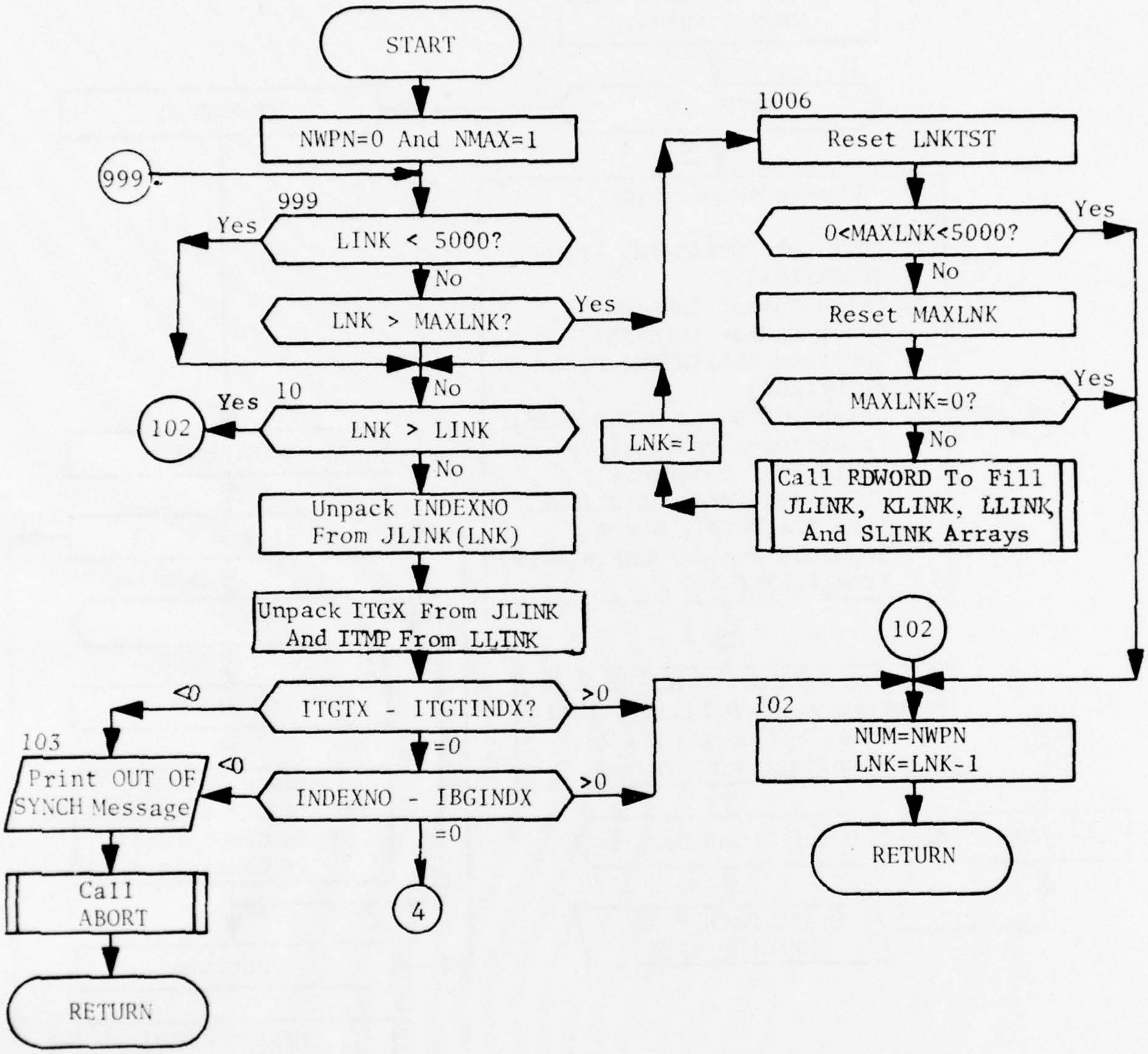


Figure 69. Subroutine UNPACKER
(Part 1 of 2)

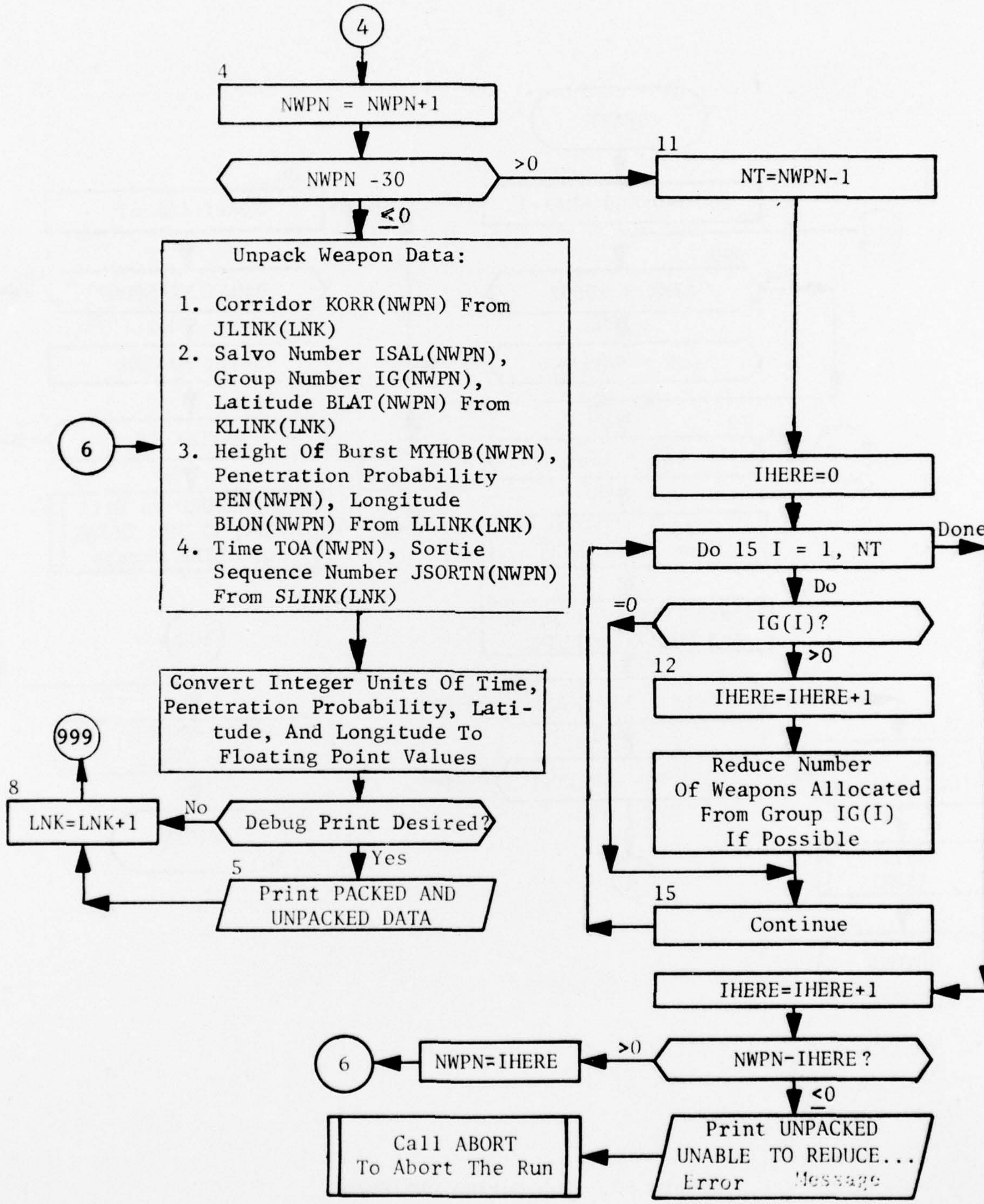


Figure 69. (Part 2 of 2)

THIS PAGE INTENTIONALLY LEFT BLANK

4.16 Subroutine WPNMODIF

PURPOSE: To allow the user to modify reliability REL and circular error probability CEP by weapon class or type and DBL probability and weapon yield by weapon group

ENTRY POINTS: WPNMODIF

COMMON BLOCKS: P, OPT, WAROUT, CWPNMOD

SUBROUTINES CALLED: PAGESKP

CALLED BY: EVAL2

Method:

WPNMODIF allows the user to modify specified weapon parameters, namely, reliability (REL), circular error probability (CEP), DBL probability, and weapon YIELD for chosen weapon types. The type name may be ALLWPNS, BOMBERS, MISSILES, a specific type name such as B-58-E, or GROUP. The parameters DBL and YIELD can only be changed by the type name GROUP. WPNMODIF reads the user's weapon parameter modifications one card at a time and multiplies the parameter by the factor on the card for the specified weapon type, class, or group. Internal checks are made by WPNMODIF to ensure that REL never exceeds unity. As each modification is made, it is printed out.

Exit from WPNMODIF is accomplished when it finds a blank card or a RETURN card at the end of the modifications.

Subroutine WPNMODIF is illustrated in figure 70.

DISTRIBUTION

Addressee	Copies
CCTC Codes	
Technical Library (C124)	3
C124 (Stock)	6
C313	1
C314	12
C520	1
C730	1
DCA Codes	
205	1
R121A	1
EXTERNAL	
Chief, Studies, Analysis and Gaming Agency, OJCS ATTN: SFD, Room 1D957, Pentagon, Washingon, DC 20301	5
Chief of Naval Operations, ATTN: OP-96C4, Room 4A478, Pentagon, Washington, DC 20350	2
Commander-in-Chief, North American Air Defense Command ATTN: NPXYA, Ent Air Force Base, CO 80912	2
Commander, U. S. Air Force Weapons Laboratory (AFSC) ATTN: AFWL/SAB, Kirtland Air Force Base, NM 87117	1
Commander, U. S. Air Force Weapons Laboratory (AFSC) ATTN: AFWL/SUL (Technical Library), Kirtland Air Force Base, NM 87117	1
Director, Strategic Target Planning, ATTN: (JPS), Offutt Air Force Base, NE 68113	2
Defense Documentation Center, Cameron Station, Alexandria, VA 22314	12
	51

THIS PAGE INTENTIONALLY LEFT BLANK

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CSM MM 9-74, Volume III	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK), Program Maintenance Manual, Weapon Allocation Subsystem		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard L. Page Angelo M. Pelliciotto Dale J. Sanders		8. CONTRACT OR GRANT NUMBER(s) DCA 100-73-C-0058
9. PERFORMING ORGANIZATION NAME AND ADDRESS System Sciences, Incorporated 4720 Montgomery Lane Bethesda, Maryland 20014		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Command and Control Technical Center Room BE-685, The Pentagon, Washington, DC 20301		12. REPORT DATE 1 June 1974 13. NUMBER OF PAGES 508
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

War Gaming, Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The CCTC Quick-Reacting General War Gaming System (QUICK) is used to generate missile and bomber plans suitable for strategic war gaming, and to simulate execution of the planned events.

This is one of five volumes describing the computer programming specifications for the Quick-Reacting General War Gaming System (QUICK). This volume addresses programs of the QUICK Weapon Allocation Subsystem, and is intended to serve as the basis for program maintenance activities. Accordingly, Volume I describes

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

243 000

20. (ABSTRACT)(Cont'd)
the program functions and contains flow charts for each program and subprogram of the Weapon Allocation Subsystem. The associated program listings which are dynamic and voluminous are not contained herein. However, the program listings may be obtained by arrangement with the CCTC QUICK Project Officer.

QUICK is documented extensively in a set of Computer System Manuals (Series 9-74) published by the Command and Control Technical Center (CCTC), Defense Communications Agency (DCA), The Pentagon, Washington, D.C. 20301.